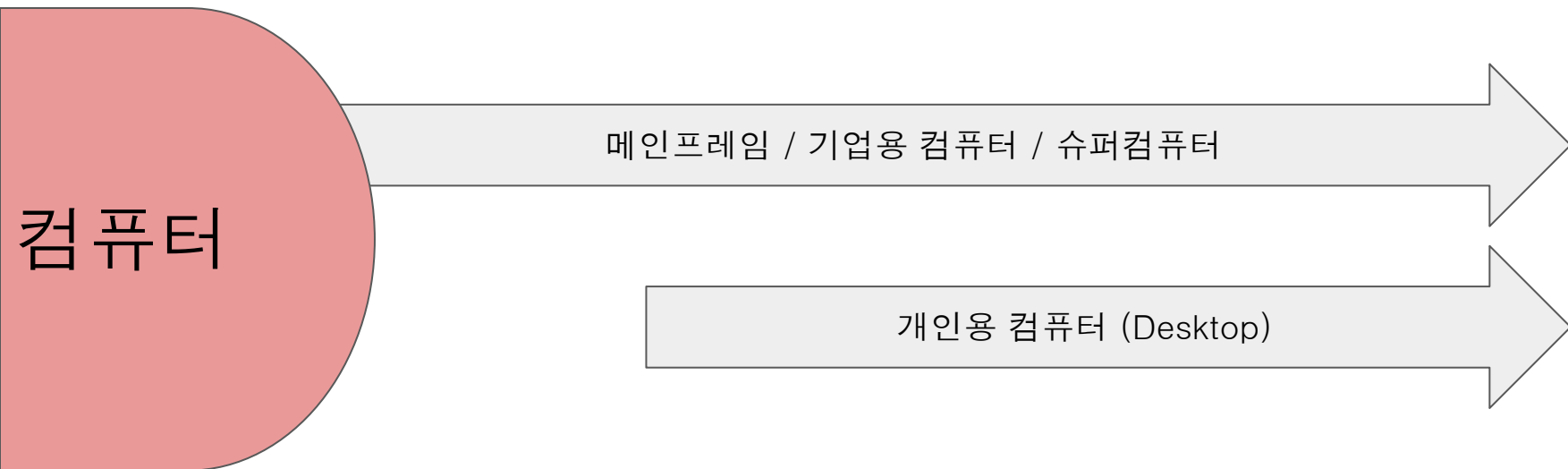
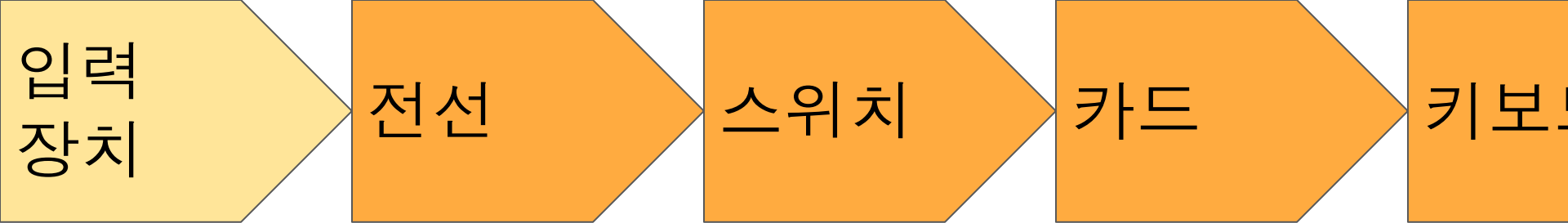
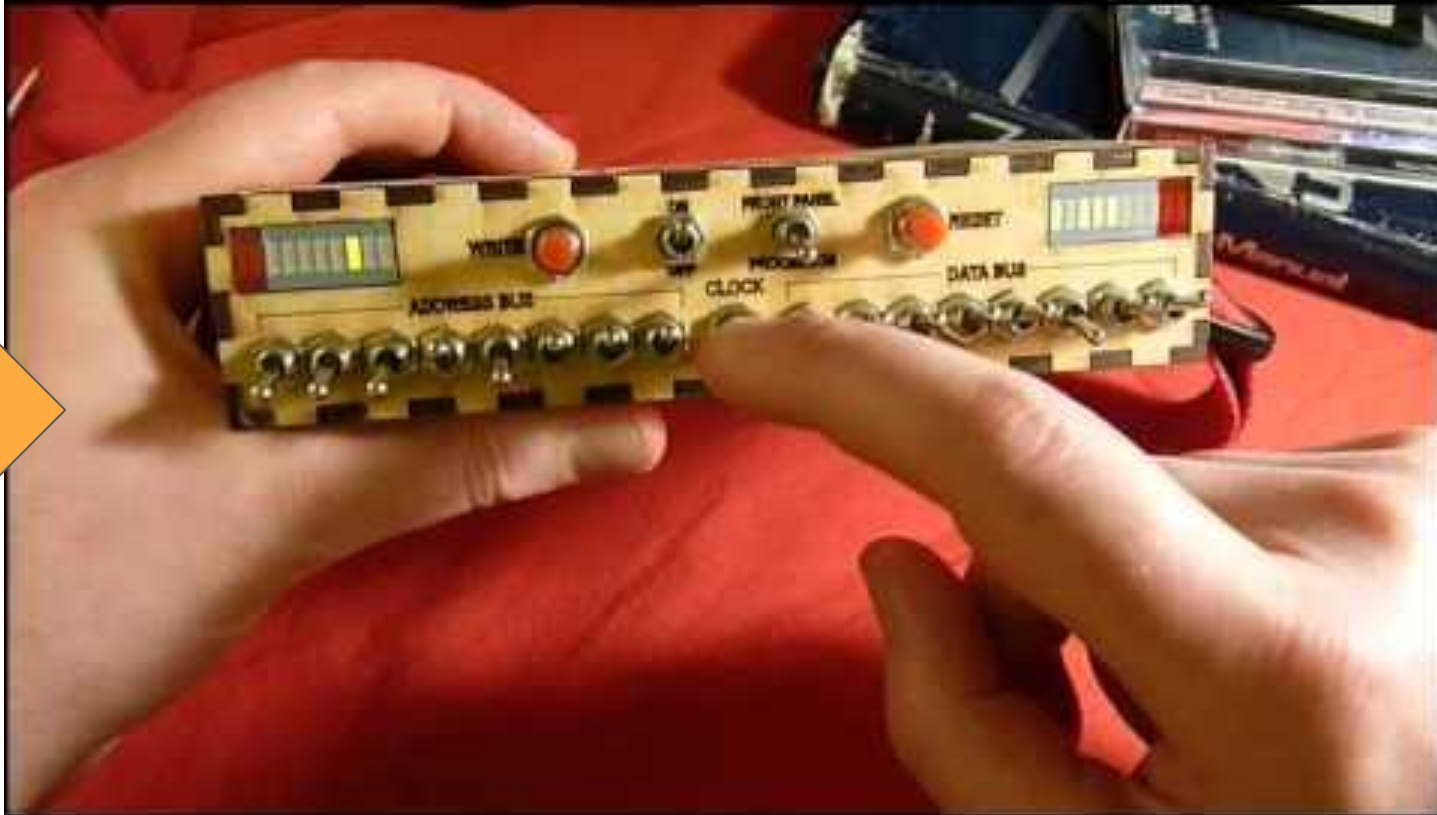


# 파 이 션

임성국 ([eventia@gmail.com](mailto:eventia@gmail.com))

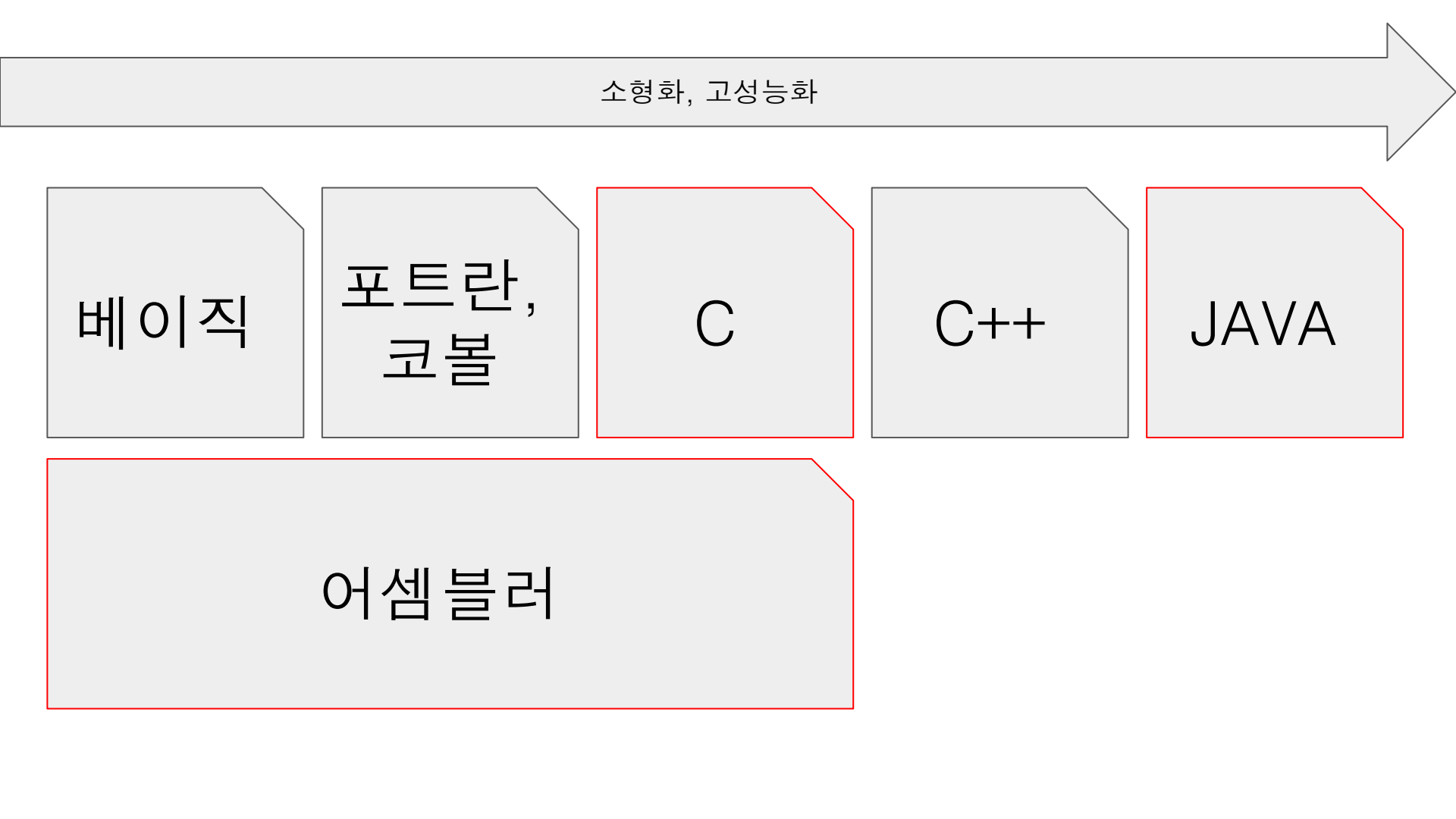


# 기계어, 어셈블러



[z80 toggle switch  
computer](#)

소형화, 고성능화



베 이직

포트란,  
코볼

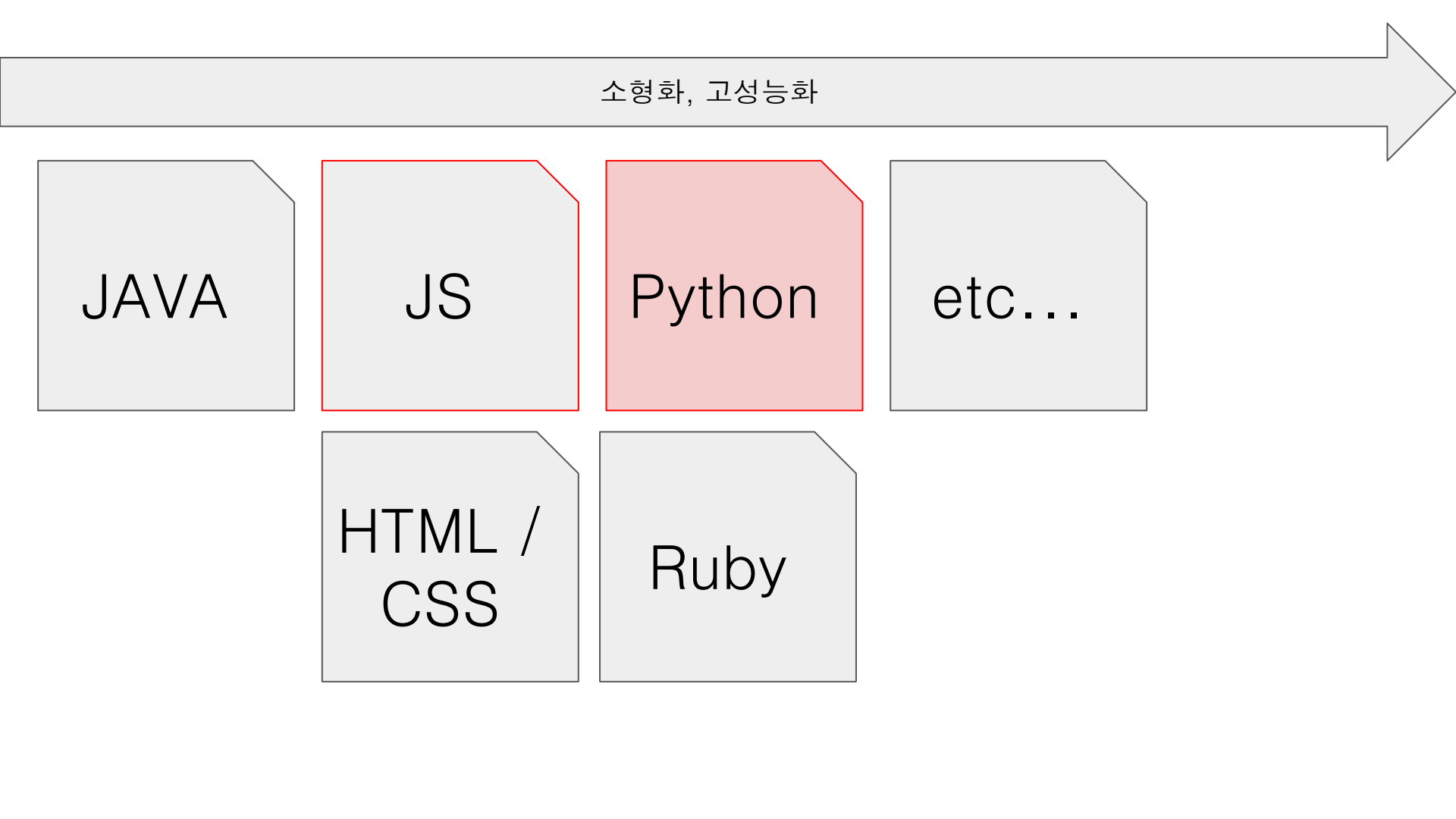
C

C++

JAVA

어셈블러

소형화, 고성능화



JAVA

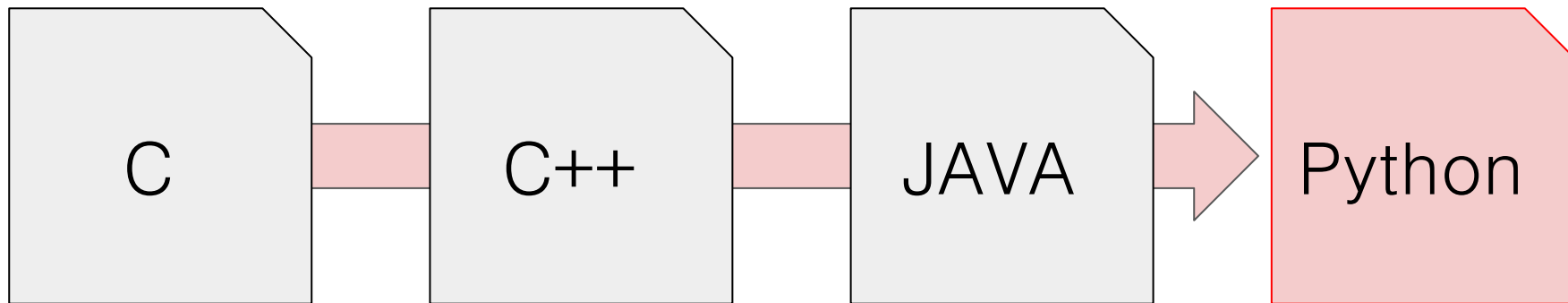
JS

Python

etc...

HTML /  
CSS

Ruby



1972년  
켄 톰슨,  
데니스 리치

1983년  
비야네  
스트롭스트롭

1991년  
제임스 고슬링

2000년  
Python 2.0  
귀도 반 로섬



# Python 01

# 설치

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com



Everybody in this country  
should learn to program  
a computer, because  
**it teaches you how to think**

- Steve Jobs-



# 파이썬이란



**Guido Van Rossum**

- 1990년 암스테르담의 귀도 반 로섬(Guido Van Rossum)이 개발한 인터프리터언어
- "몬티 파이썬의 날아다니는 서커스(Monty Python's Flying Circus)"
- 크리스마스에 심심해서 시작한 개발
- Life is short. You need Python

# 파이썬이란

- 구글 소프트웨어 작성에 큰 비중
- 해외 교육용 언어로 사용
- 드롭박스(DropBox), 장고(Django)
- 공동작업, 유지보수가 쉬운 언어
- 다른 언어(C, C++) 로 작성된 많은 모듈이 공개
- 데이터처리, TensorFlow

```
print("Hello World")
```

# You Need ...

## → 파이썬

- ◆ 최신 버전 - 2 의 파이썬 다운, 설치
- ◆ 3.XX.Y -> 3.(XX-2).Max

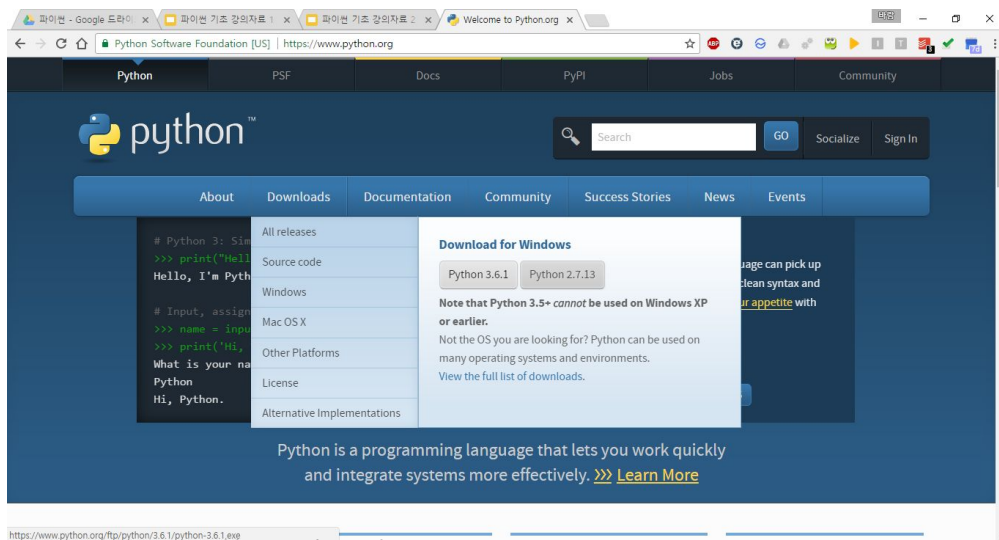
## → Visual Studio Code

- ◆ 텍스트 에디터

# 파이썬 설치

- python.org 에서

다운로드



- 설치시 **C:\Python\Python3xx\** 에 설치

- path** 부분을 제크 [v] 할 것



# 처음코딩 - Python 02

## 배경지식

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

# (윈도우 기준)파이썬을 실행시키는 3가지 방법

1. 명령 프롬프트에서 `python.exe` 실행
2. IDLE
3. 에디터로 코드작성 후 실행

# 메모리와 변수



# 결과 예상해보기

```
>>> teacher = "Lim"  
>>> print (teacher)
```

```
>>> a = 7  
>>> b = 5  
>>> print (a+b)
```

```
>>> a = 7  
>>> b = 5  
>>> print ("a+b")
```

# 결과 예상해보기

```
>> teacher = "LIM"  
>> print (teacher )  
LIM
```

```
>> a = 7  
>> b = 5  
>> print (a+b)  
12
```

```
>> a = 7  
>> b = 5  
>> print ("a+b")  
a+b
```

teacher = “LIM”

의 의미는?

- teacher = "LIM"

- 변수 teacher 에 "LIM" 을 넣어라

# print(a+b) 와 print("a+b") 의 차이

**print** (a+b)

- a 라는 변수에 있는 값과 b 라는 변수에 있는 값을 더해서 화면에 출력하라는 의미

**print** ("a+b")

- "a+b" 값을 화면에 출력하는 의미

# 변수 (Variable)

- 메모리(RAM)에 위치하고,  
특정한 값을 저장할 수 있는 장소
- 저장된 위치값인 주소(address)를 가짐

# 변수 (Variable)



메모리 위주소

Memory

Address

Variable

"LIM"
7
3

0x0007

0x0006

0x0005

0x0004

0x0003

0x0002

0x0001

teacher

b

a

## 변수 이름 쓰기

- 알파벳, 숫자, 언더스코어(\_) 사용 가능  
ex) data = 0, \_a12 = 2, \_gg = 'afdf'
- 첫글자는 영어 알파벳으로 사용
- 의미 있는 단어로 표기  
ex) teacher\_name = 'LIM'
- 대소문자 구분  
ex) ABC와 Abc는 같지 않다
- 프로그램에서 이미 사용하고 있는 예약어 사용 금지  
ex) for, if, else 등



# 간단한 연산

- 복잡한 프로그램을 작성하기 앞서 간단한 사칙연산과 문자열 처리 등의 기본을 배워야 함
- 이를 위해 본 장에서는 아래 내용을 습득
  - 1) 기본 자료형 (Data Types)
  - 2) 연산자와 피연산자
  - 3) 데이터 형변환
- 이를 통해 간단한 프로그램 작성의 기초를 익힘

# 기본 자료형 (Data Type)

유형			설명	예시	선언 형태
수치자료형	정수형	Integer	양/음의 정수	1,2,3,100, -9	data = 1
	실수형	Float	소수점이 포함된 실수	10.2, -9.3, 9.0	data = 9.0
문자형(문자형)		String	따옴표 ('ad')에 들어가 있는 문자형	abc, a20abc	data = 'abc'
논리/불린 자료형		Boolean	참 또는 거짓	True, False	data = True

# 변수와 기본자료형

```
>>> a = 1 # Integer
>>> b = 1 # Integer
>>> print (a, b)
1 1
>>> a = 1.5 # Float
>>> b = 3.5 # Float
>>> print (a, b)
1.5 3.5
>>> a = "ABC" # String
>>> b = "101010" # String
>>> print (a, b)
ABC 101010
>>> a = True # Boolean 대소문자 구분
>>> b = False # Boolean 대소문자 구분
>>> print (a, b)
True False
```

# 연산자와 피연산자

- $+$ ,  $-$ ,  $*$ ,  $/$  같은 기호들을 연산자
- 연산자에 의해 계산이 되는 문자나 숫자를 피연산자
- $3 + 2$  에서 3과 2는 피연산자,  $+$ 는 연산자임
- 수식에서 연산자의 역할은 수학에서 연산자와 동일
- 연산의 순서는 수학에서 연산 순서와 같음
- 문자간에도  $+$  연산이 가능함

# 제곱승과 나머지 연산자 : \*\* %

## \*\* 는 제곱승 연산자

```
>>> print (3 * 3 * 3 * 3 * 3) # 3을 다섯 번 곱함
243
>>> print (3 ** 5)           # 3의 5승
243
```

Python Shell

## % 는 나머지 연산자

```
>>> print (7 / 2)             # 7 나누기 2 (정수형 계산)
3.5
>>> print (7 % 2)             # 7 나누기 2의 나머지는
1
```

Python Shell

증가, 감소연산 +=, -=, \*=, /=, (파이썬 X)++, --

```
>> number = 10  
>> print(number)
```

```
>> number = number + 1  
>> number += 1  
>> print(number)
```

hello world

## 결과 예상해보기

다음 내용을 파일 `hello.py` 에 저장

```
print("hello world")
```

명령어입력창에서 다음 명령 입력

```
>> python hello.py
```



# 간단한 구조



# input() 함수

```
answer = input()
```



# 처음코딩 - Python 03

## 주석, 상수, 문자열

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

# 문자 뒤로 그 줄의 끝까지의 짧은 문장

# 주석 예문

```
print('hello world') # 시작하는 프로그램코드 hello world  
# 코드가 어떻게라는 물음에 답한다면 주석은 왜라는 물음의 답이다.
```

- 숫자, 문자열
- 프로그램내에 정해진 형태로 지정됨
- 한번 지정된 것을 변하지 않음

5

1.23

'This is a book'

"Sting"

→ 정수형 (int)

◆ 513

→ 부동 소수점 숫자형 (float)

◆ 3.23

◆ 5.3E-2



```
import sys
```

```
t1 = sys.maxsize
```

```
t2 = t1+1    #파이썬 3.x 에서는 long, int 통합
```

```
t3 = t2**10   # 큰 수도 O.K.
```

```
print(t1)
```

```
print(t2)
```

```
print(t3)
```

```
print(type(t1))
```

```
print(type(t2))
```

```
print(type(t3))
```

→ 작은 따옴표

◆ '바람이 부는 날'

→ 큰 따옴표

◆ "보고 싶은 영화"

◆ "보고 싶은 영화는 '모던타임즈'다"

➔ 여러줄 문자열

""""나 하늘로 돌아가리라  
새벽빛 와 닿으면 스러지는  
이슬 더불어 손에 손을 잡고""""

→ 상수는 수정할 수 없음

# 문자열 포맷 format()

문자열에 어떤 정보를 포함시킬때 사용

'...{0} ... {1}...' .format( name , age )

문자열

Dot - 연결

format() 함수

# 문자열 포맷 format() 예문

```
age = 26
```

```
name = 'Python'
```

```
print('{0} was {1} years old'.format(name, age))
```

```
print('{0} is easy'.format(name))
```

# 문자열 더하기

## → 문자열 더하기

◆ `name + ' 나이는 ' + str(age) + ' 이다'`

## → `format()` 이용

◆ `'{0} 나이는 {1} 이다'.format(name, age)`

## format() 용법 추가

- 소수점 이하 원하는 자리까지 표기

```
>>> print("{0:.3f}".format(1/3))
```

- 밑줄로 11칸 채우고 가운데 정렬

```
>>> print("{0: ^11}".format("Hello"))
```

- ➔ 사용자 지정 키워드 사용

[illegible]



# f-string

→ 문자열 앞에 f 를 붙여 사용

```
>>> month = 3
```

```
>>> print(f"2025년 {month}월")
```

# f-string

py# f-string 왼쪽 정렬

```
s1 = 'left'
```

```
result1 = f'|{s1:<10}|'
```

```
print(result1)
```

# f-string 가운데 정렬

```
s2 = 'mid'
```

```
result2 = f'|{s2:^10}|'
```

```
print(result2)
```

# f-string 오른쪽 정렬

```
s3 = 'right'
```

```
result3 = f'|{s3:>10}|'
```

```
print(result3)
```

[결과]

```
|left      |
```

```
|  mid     |
```

```
|    right|
```

# print() 와 개행(\n)

→ print() 마지막 \n 는 자동

→ 줄바꿈을 막으려면,

```
>>> print("Hello World", end="")
```

# Escape 문자

- ❑ 특수문자 표기 방법

- ❑ \ + 특수문자

- ❑ 자주 사용하는 특수문자

- ❑ ' 작은 따옴표

- ❑ " 큰 따옴표

- ❑ \ 역슬래시

- ❑ \n 개행문자

- ❑ \t 탭문자

- ❑ \ 여러줄

# Escape 문자 표시

- ❑ `print("나 하늘로 돌아가리라 \n새벽빛 와 닿으면 스러지는 \n이슬  
더불어 손에 손을 잡고 ")`
- ❑ `print("나 하늘로 돌아가리라 \n새벽빛 와 닿으면 \  
스러지는 \n이슬\  
더불어 손에 손을 잡고 ")`

# 순(Raw)문자열

- ❑ 문자열 내부의 이스케이프 문자를 처리하지 않고 그대로 출력
- ❑ `r` 또는 `R` 을 문자열 앞에 붙여 표시
  - ❑ `r"새로운 행을 만드는 기호는 \n 입니다."`
  - ❑ `print(r"새로운 행을 만드는 기호는 \n 입니다.")`



# 처음코딩 - Python 04

## 변수, 상수

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

→ 정보를 담고 수정할 수 있는 공간



- ❖ 영어 알파벳 대문자, 소문자, 숫자, 그리고 특수기호로 밑줄 (\_)만 사용
- ❖ 첫 문자는 알파벳 (A-Z,a-z)이거나 밑줄 (\_)

## 좋은 이름, 나쁜 이름

❑ i ..... [ ]

❑ name ..... [ ]

❑ 2things ..... [ ]

❑ my-name ..... [ ]

❑ aaaa ..... [ ]

# 변수와 리터럴 상수 사용

```
number = 5
```

```
print(number)
```

```
number = number+1
```

```
print(number)
```

```
content1 = """인공지능을 도입한 바둑프로그램의 개발이 가속화되고 있다.  
구글의 딥마인드로 대표..."""
```

```
print(content1)
```

# 여러줄 문자열 처리

```
content1 = """인공지능을 도입한 바둑프로그램의 개발이 가속화되고 있다.  
구글의 딥마인드로 대표..."""
```

```
content2 = """인공지능을 도입한 바둑프로그램의 개발이 가속화되고 있다.\n구글의 딥마인드로 대표..."""
```

```
content3 = """인공지능을 도입한 \n바둑프로그램의 개발이 \n가속화되고 있다.\n\n구글의 딥마인드로 대표...\n"""
```

```
print(content1)  
print(content2)  
print(content3)
```

# 논리적/물리적 명령행

→ 물리적 명령행

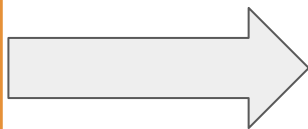
◆ 프로그램 코드내에 직접 표현된 한줄

→ 논리적 명령행

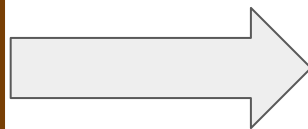
◆ 파이썬 인터프리터가 한줄로 인식하는 행

# 논리적 명령행

```
i = 5  
print(i)
```



```
i = 5;  
print(i) ;
```



```
i = 5;print(i) ;
```

; (세미콜론) 을 사용하여 물리적 명령행과 다른 논리적 명령행을 만들어 사용하지  
말것 !

- ❑ 파이썬은 들여쓰기 (indent) 가 중요
- ❑ 들여쓰기는 새 블록 시작을 의미
- ❑ 같은 들여쓰기는 동일 블록
- ❑ 공백 4개로 통일



# 처음코딩 - Python 05

## 기본연산자

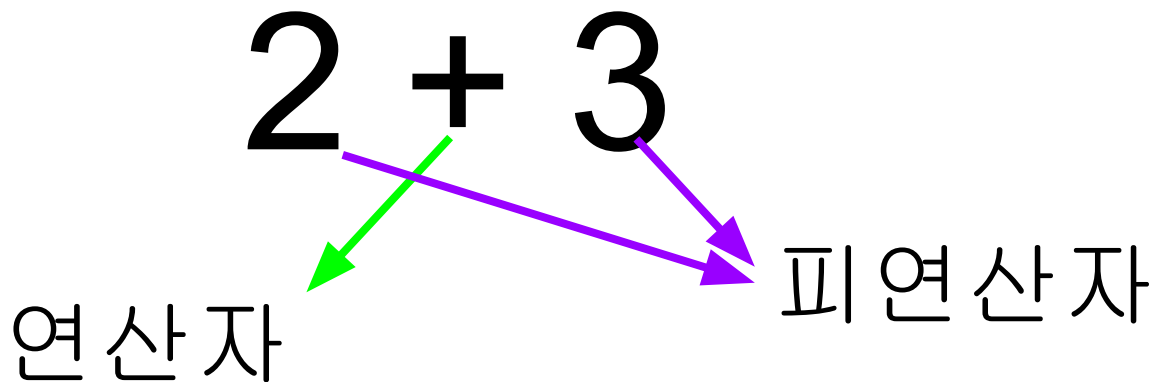
바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com



# 연산자

→ 계산, 처리할 때 쓰이는 기능



# 기본 연산자 1

+ (덧셈연산자)

두 객체를 더한다.

숫자는 값을 더하고, 문자열은 연결된다.

- (뺄셈연산자)

첫번째 숫자에서 두번째 숫자를 뺀다.

\* (곱셈연산자)

두 숫자의 곱이 결과가 된다.

반복된 문자열을 만든다.

\*\* (거듭제곱연산자)

x 의 y 제곱을 반환한다.

## 기본 연산자 2

/ (나눗셈연산자)

x 를 y로 나눈 값을 만든다.

// (정수형 나눗셈연산자)

x 를 y로 나눈 몫을 만든다.

% (나머지연산자)

x 를 y 로 나눈 나머지를 만든다.

<< (왼쪽시프트연산자)

지정된 수만큼 왼쪽으로 비트 단위로 이동한다.

>> (오른쪽시프트연산자)

지정된 수만큼 오른쪽으로 비트 단위로 이동한다.

# 기본 연산자 3

& (비트AND연산자)

비트단위의 AND 연산결과를 보인다.

| (비트OR연산자)

비트단위의 OR 연산결과를 보인다.

^ (비트XOR연산자)

비트단위의 XOR 연산결과를 보인다.

~ (비트반전연산자)

비트를 반전해서 만들어진 2의 보수값  $-(x+1)$  을 만든다.

# 기본 연산자 4

< (작다)

작으면 True 를, 아니면 False 를 만든다.

> (크다)

크면 True 를, 아니면 False 를 만든다.

<= (작거나 같다)

작거나 같으면 True 를, 아니면 False 를 만든다.

>= (크거나 같다)

크거나 같으면 True 를, 아니면 False 를 만든다.

!= (같지 않다)

같으면 True, 아니면 False 를 만든다.

# 기본 연산자 5

not (불리언 NOT 연산자)

True는 False를, False는 True를 만든다.

and (불리언 AND 연산자)

2개 피연산자가 모두 True일때만 True를 만든다. 나머지는 False

단축계산 : x and y 에서 x 가 False 면 y 를 계산하지 않음.

or (불리언 OR 연산자)

2개 피연산자가 모두 False일때만 False를 만든다. 나머지는 True

단축계산 : x or y 에서 x 가 True 면 y 를 계산하지 않음.



# 처음코딩 - Python 06

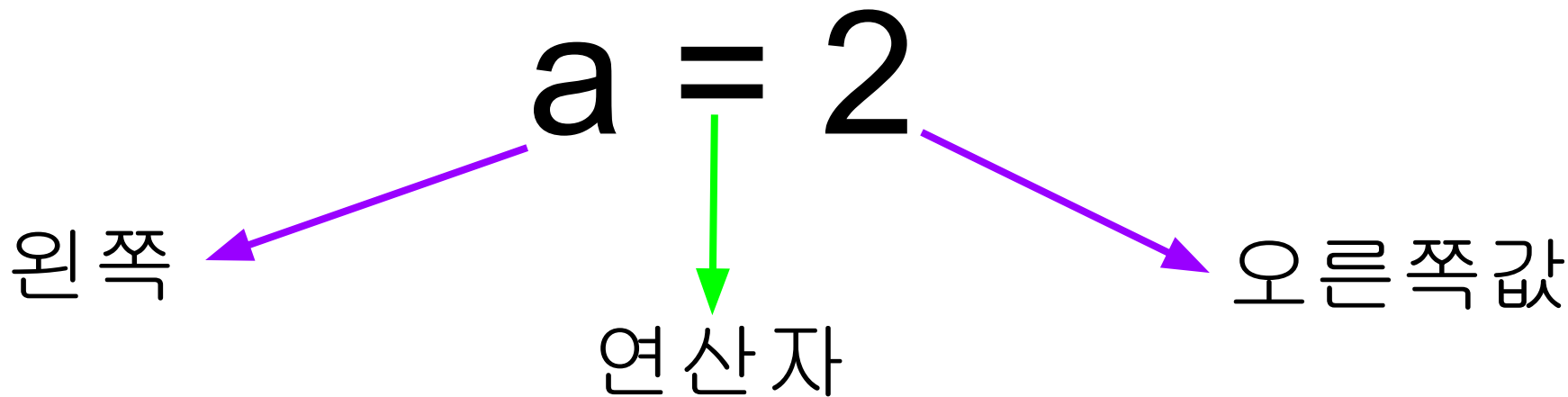
## 할당연산자

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

# 대입연산자, 할당연산자, Assignment Operators

→ 오른쪽의 값을 왼쪽(변수)에 넣어줌





# 할당연산자 사용법 1

**a = 2**

**a = a \* 3**

**a = 2**

**a \*= 3**

**a = a+1**

a += 1 ----- (O)

a++ ----- (X)

# 할당 연산자와 복소수 1

$$j = 2$$

$$j = j$$

$$j = 2j$$

$$j = 2*j$$

**a = 1-2j**

**b = complex(1,-2)**

# 복소수 연산 1

$$a*b$$

$$a+b$$

$$a-b$$

$$a/b$$

## 복소수 연산 2

`a.real`      # 실수부

`a.imag`      # 허수부

`a.conjugate()`      # 켤레복소수

`abs(a)`      # 크기



# 처음코딩 - Python 07

## 연산 순서

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com



# 연산자 우선순위 (파이썬 레퍼런스 매뉴얼)

낮음



높음

Operator	Description
<code>lambda</code>	Lambda expression
<code>if - else</code>	Conditional expression
<code>or</code>	Boolean OR
<code>and</code>	Boolean AND
<code>not x</code>	Boolean NOT
<code>in, not in, is, is not, &lt;, &lt;=, &gt;, &gt;=, !=, ==</code>	Comparisons, including membership tests and identity tests
<code> </code>	Bitwise OR
<code>^</code>	Bitwise XOR
<code>&amp;</code>	Bitwise AND
<code>&lt;&lt;, &gt;&gt;</code>	Shifts
<code>+, -</code>	Addition and subtraction
<code>*, @, /, //, %</code>	Multiplication, matrix multiplication division, remainder <a href="#">[5]</a>
<code>+x, -x, ~x</code>	Positive, negative, bitwise NOT
<code>**</code>	Exponentiation <a href="#">[6]</a>
<code>await x</code>	Await expression
<code>x[index], x[index:index], x(arguments...), x.attribute</code>	Subscription, slicing, call, attribute reference
<code>(expressions...), [expressions...], {key: value...}, {expressions...}</code>	Binding or tuple display, list display, dictionary display, set display

# 연산 순서 변경

❏ 괄호 ( ) 를 사용하여 순서 변경

- $\text{result} = 1 + 2 * 3$

- $\text{result} = (1 + 2) * 3$

## 연산자가 같은 우선순위에 있을때

❑ [기본] 왼쪽에서 오른쪽으로

$$1+2+3$$

$$(1+2)+3$$

❑ [할당] 오른쪽에서 왼쪽으로

$$a = b = c$$

$$a = (b = c)$$

# 연산자 사용 예제

1. 사각형의 두변의 길이를 입력하면 면적과 둘레를 구하는 프로그램을 작성하시오. 이때 두변은 length, width 변수를 쓰고, 각각 52 와 379 를 넣어서 프로그램을 작성합니다.

최종 결과화면은 다음과 같이 나오게 합니다. {숙제1}

2. 두변을 input() 으로 입력받아 면적과 둘레 출력

면적은 \*\*\*\* 입니다.

둘레는 \*\*\*\* 입니다.

# 연산자 사용 예제 소스

```
length = 5.2
```

```
width = 3.79
```

```
area = length * width
```

```
perimeter = (length + width) * 2
```

```
print("면적은 ", area, "입니다.")
```

```
print("둘레는 ", perimeter, "입니다.")
```

```
D:\DEV\PyWork\zbc_python>python expression.py
면적은 19.708000000000002 입니다.
둘레는 17.98 입니다.
```

$$64 \text{ bits} = 1 + 11 + 52$$

1 : 부호 표시

11 : 지수 표시 ( $2^{11}$ , 11비트)

52 : 수자 표시 ( $2^{52}$ , 52비트)

# 0.1

$$0.1 = \square 2^{-1} + \square 2^{-2} + \square 2^{-3} + \square 2^{-4} + \square 2^{-5} + \square 2^{-6} + \square 2^{-7} + \dots$$

0.1(10진수)

= 0.000110011001100110011001100110011... (2진수)

0.1+0.2

0.1 과 0.2 를 더해보고 결과를 살펴보세요.

0.1 과 0.2 를 더한 결과가 0.3 입니까?

>> format, var, '.20f' 사용



# 0.1+0.2

```
a = 0.1  
b = 0.2  
var = a + b  
print('var = {0:.20f}'.format(var))  
print(f'var = {var:.20f}')
```



# 처음코딩 - Python 08

## if 조건문

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

1. 순차구조 (위에서 아래로 하나씩 순서대로) - 객체지향, 스크래치, 엔트리
2. 조건구조 (if, switch-case)
3. 반복구조 (for, while)

# 조건문이란?

□ 조건에 따라 정해진 명령을 실행

# 실생활속 조건문

❑ 앞차와의 거리가 10m 이하**면**

 속도를 10km/h 로 줄여라.

❑ 암호입력이 3회 실패하**면**

 3분간 대기 상태로 만들어라.

❑ 컵라면에 물을 부은지 3분이 지나**면**

 먹어라.

# 파이썬의 조건문

- ❑ 옳은지 틀린지를 분명히 알수 있는 **조건**과
- ❑ 각각의 경우에 실행할 **명령**들로 구성

**if 조건 :**

(띄워쓰기) 명령

REMEMBER :

- ❑ 4칸 띄워쓰기 (indent)

- ❑ 콜론 : 은

새로운 블록의 시작

# if 문 예제 1

❑ if 조건문을 사용해서 사용자가 입력한 숫자가 파이썬에 저장된 수보다 크면 "틀렸습니다. 조금 더 작은수입니다." 를, 작으면 "틀렸습니다. 조금 더 큰 수입니다." 를, 같으면 "맞았습니다."를 출력하는 프로그램을 만들어 보세요.

if, elif, else 사용 / input() 함수 사용 / print() 함수 사용 / 비교연산자  
==, <, > 사용 / 대입연산자 = 사용 / 변수 사용 / 비교할 값은 23



# if .. elif .. else 문

명령

if 조건 :

명령

elif 조건 :

명령

else :

명령

명령

REMEMBER :

- 4칸 띄워쓰기 (indent)
- 콜론 : 은 새로운 블록의 시작
- elif / else 블록은 생략 가능

if (배가고프다) :

    라면을먹는다

커피를마신다.

# input( ) 함수

- ❑ **input**("화면에 표시할 내용")
  - ❑ 결과값은 문자열
- ❑ 문자열을 정수로 변환
  - ❑ **int**(문자열)

## input( ) 함수 사용예

```
name = input("이름을 입력하세요")
```

```
number = int(input("수를 입력하세요"))
```

## if 문 예제 2

- ❑ if 조건문을 사용해서 사용자가 입력한 숫자가 짝수이면, "짝수입니다." 를 출력하고, 3의 배수이면, "3의 배수입니다." 를 출력하는 프로그램을 만들어 보세요.

if 조건문 사용 / input() 함수 사용 / print() 함수 사용 / 나머지 연산  
% 사용 / 대입연산자 = 사용 / 변수 사용



# 처음코딩 - Python 09

## while 문

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

# while 반복문이란?

- ❑ 특정조건이 참(True)인 경우 블록안의 명령들을 반복해서 실행
- ❑ else 절이 따라올 수 있음

# while 기본 구문

**while** 조건:

명령1

명령2

명령3



# while 기본 구문

**while** 조건:

명령

**else :**

명령

# while 기본 구문

**while** 조건:

명령

**else :**

명령

}



else 가 사용되는 경우는  
continue, break 에서 설명

## while 문 예제 1

- ❑ 반복적으로 사용자가 입력한 숫자가 저장된 수보다 크면 "틀렸습니다. 조금 더 작은수입니다." 를, 작으면 "틀렸습니다. 조금 더 큰 수입니다." 를 출력하고, 같으면 "맞았습니다."를 출력하고 종료하는 프로그램을 만들어 보세요. (저장된 수는 23)

## while 문 예제 2

□ 사용자가 입력한 숫자가 짝수이면, "짝수입니다." 를 출력하고, 3의 배수이면, "3의 배수입니다." 를 한줄에 출력하는 프로그램을 만들어 보세요. 단, 777을 입력하면 종료하고, 나머지에 대해서는 반복해서 출력합니다.

{숙제}



# 처음코딩 - Python 10

## for .. in 문

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

1. 순차적 구조 (위->아래, 순서대로 실행)
2. 조건문 (특정 조건에서만 실행되는 구간, 블록)
3. 반복문 (조건 혹은 횟수를 정해서 실행되는 구간, 블록)
  - for
  - while

- ❑ for .. in 문은 열거형에 포함된 각 항목을 하나씩 거쳐가며 실행
  - ❑ 열거형 : 여러 항목이 나열된 어떤 목록

## for 예제

```
for i in range(1,5):  
    print(i)  
else :    # 사용되지 않음  
    print("for문이 종료되었습니다")
```



# range() 함수

정수(음수포함)만 사용

지정되지 않으면 기본값 0

포함되지 않음

지정되지 않으면 기본값 1

range (시작, 끝, 증감)



# 처음코딩 - Python 11

## break, continue

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

- ❑ break 문은 루프를 강제로 빠져나올 때 사용
  - ❑ 루프 조건이 `False`가 되지 않았거나
  - ❑ 열거형의 끝까지 루프가 도달하지 않았지만
  - ❑ 루프 문의 실행을 강제로 정지시키고 싶을 때 사용
- ❑ break 실행시 else 블록은 실행되지 않음

## break 예제

```
while True:
    s = input('Enter something : ')
    if s == 'quit':
        break
    print ('Length of the string is', len(s))

print ('Done')
```

- ❑ continue 문은 현재 실행중인 루프 블록의 나머지 명령문들을 실행하지 않고 곧바로 다음 루프로 넘어간다

## continue 예제

```
while True:
    s = input('Enter something : ')
    if s == 'quit':
        break

    elif len(s) < 3:
        print('Too small')
        continue

    print('Input is sufficient length')
    # Do other kinds of processing here...
```



# 처음코딩 - Python 20

## 자료구조 - 리스트

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

리스트 [ 1, 2, 3 ] ==> value, index(0 부터 시작), 추가,  
삭제,수정O

튜플 ( 1, 2, 3 ) ==> value, index(0 부터 시작), 추가,  
삭제,수정X,

딕셔너리 { "apple" : "사과", "banana" : "바나나" } ==> key  
: value 의 쌍으로 구현



## 리스트란...

- ❑ 순서대로 정리된 항목들을 담고 있는 자료 구조
- ❑ 추가하거나 삭제 가능
- ❑ 대괄호 [ ] 로 묶음
- ❑ `mylist = [1, 2, 'a', 'book', ['one', 'two']]`

# 객체와 클래스 간단 소개

- ❑ `i = 5`

- ❑ `int` 라는 클래스의 객체 `i` 를 만든다

- ❑ `i` 에 5 라는 값을 저장한다.

- ❑ 클래스 - 만들고 싶은 것을 찍어낼 수 있는  
무형의 틀
- ❑ 객체 - 클래스로 찍어서 만들어 낸 실체

❑ 함수 - 클래스에 정의된 함수 => 메서드

❑ 변수 - 클래스에 정의된 변수 => 필드

# list 클래스

list 클래스

메소드(함수):  
append( )

1. list 클래스로 객체를 만든다.
2. 객체 이름을 mylist 라 한다.
3. mylist 는 list 클래스의 메소드(함수)를 쓸 수 있다.
4. list 클래스에는 append() 라는 메소드가 있다.
5. mylist.append( ) 처럼 써서 append() 메소드를 사용한다.

```
# This is my shopping list
```

```
shoplist = ['apple', 'mango', 'carrot', 'banana']
```

```
print('I have', len(shoplist), 'items to  
purchase.')
```

```
print('These items are: ', end='')
```

```
for item in shoplist:
```

```
    print(item, end=' ')
```

```
print('\nI also have to buy rice.')
```

```
shoplist.append('rice')
```

```
print('My shopping list is now', shoplist)
```

```
print('I will sort my list now')
```

```
shoplist.sort()
```

```
print('Sorted shopping list is', shoplist)
```

```
print('The first item I will buy is', shoplist[0])  
olditem = shoplist[0]  
  
del shoplist[0]  
print('I bought the', olditem)  
print('My shopping list is now', shoplist)
```



# 자료구조

## ● 리스트

리스트는 여러 요소를 담을 수 있고, 수정, 삭제가 가능합니다.  
데이터들은 [ ] 안에 담게 됩니다.

`a = [1, 2, 3, 4, 5, 6, 7, 8]`

리스트의 각 요소는 앞에서부터 인덱스 번호가 0부터 1씩 증가하면서 붙여집니다. 즉 `a[0]`은 첫 번째 요소인 1입니다. 세 번째 요소인 3은 `a[2]`입니다. 0부터 시작하기 때문에 n번째 요소의 인덱스번호는 `n-1`이 됩니다.

리스트[], 튜플(), 딕셔너리{} - 자료구조 (data structure)

변수 - 이름, 주소.... 문자열, / 숫자, 정수, 실수, 허수

학급 - 성적 ... 이름, 주소, 번호, 영어성적, 수학성적, 국어성적,  
[중학교때성적들]==> 변수

b = ["홍길동", "종로구 1번지", 12, 50, 45, 46, [55, 45, 66] ]

a = [1, 2, 3, 4, 5, 6, 7, 8]

0 1 2 3 4 5 6 7

-8 -7 -6 -5 -4 -3 -2 -1

b[3], b[4], b[5] ==> 50, 45, 46 /// b[3:6] (이상, 미만) => [50, 45, 46]

# 자료구조

## ● 인덱스, 슬라이싱

**a = [1, 2, 3, 4, 5, 6, 7, 8]**

인덱스 번호를 이용해 특정 위치값을 읽거나 수정할 수 있습니다.

**a[0]            # 1**

인덱스 번호의 시작과 끝을 지정하면 사이의 값을 읽어옵니다.

**a[0:3]        # [1,2,3]    - 0번부터 시작(이상) 3번 전까지(미만)**

**a[3:-2]      # 3부터 -2전까지, -2는 뒤에서 두번째**

```
a = [5, 10]
```

```
del a[1]          # a = [5]
```

```
a.append(15)      # a = [5, 15]
```

```
a.pop()           # a = [5]
```

```
a.append(8)       # a = [5, 8]
```

```
a.insert(1, 11)   # a = [5, 11, 8]
```

```
dir(a)            # 리스트 a 가 가지고 있는 메소드를 보여줌
```

```
range(10)         # 객체 10 ... 0 시작 [0, 1, 2, ... , 9] range 객체
```

```
list(range(10))    # 리스트 변환 [0, 1, 2, ... , 9]
```

## Q. 친구들의 리스트 생성하기

제일 친한 친구 5명의 이름을 리스트에 저장했다가 출력하는 프로그램을 작성하자

```
친구의 이름을 입력하시오: 홍길동  
친구의 이름을 입력하시오: 강감찬  
친구의 이름을 입력하시오: 이순신  
친구의 이름을 입력하시오: 권율  
친구의 이름을 입력하시오: 정약용  
['홍길동', '강감찬', '이순신', '권율', '정약용']
```

sol.

```
friend_list = []

friend = input("친구의 이름을 입력하시오: ")
friend_list.append(friend)

friend = input("친구의 이름을 입력하시오: ")
friend_list.append(friend)

friend = input("친구의 이름을 입력하시오: ")
friend_list.append(friend)

friend = input("친구의 이름을 입력하시오: ")
friend_list.append(friend)

friend = input("친구의 이름을 입력하시오: ")
friend_list.append(friend)

print(friend_list)
```



# 처음코딩 - Python 21

## 자료구조 - 튜플

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

## ○ 튜플

**b = (1, 2, 3, 4, 5, 6, 7, 8)**

튜플은 리스트와 달리 수정할 수 없습니다. 한번 만들어진 튜플은 상수처럼 읽기만 가능합니다.

인덱스, 슬라이싱 모두 리스트와 같은 방식으로 사용됩니다.



- ❑ 여러 객체를 모아 담는데 사용
- ❑ 수정 불가
- ❑ 문자열과 같이 비정적인 객체 담음
- ❑ 튜플은 생략할 수 있는 괄호로 묶고, 쉼표로 구분된 여러 개의 항목으로 정의
- ❑ `zoo = ('python', 'elephant', 'penguin')`

## 튜플 tupdemo.py

```
zoo = ('python', 'elephant', 'penguin')
print('Number of animals in the zoo is', len(zoo))

new_zoo = 'monkey', 'camel', zoo
print('Number of cages in the new zoo is', len(new_zoo))

print('All animals in new zoo are', new_zoo)
print('Animals brought from old zoo are', new_zoo[2])
print('Last animal brought from old zoo is', new_zoo[2][2])
print('Number of animals in the new zoo is', \
      len(new_zoo)-1+len(new_zoo[2]))
```

# 튜플 tupdemo.py 결과

Number of animals in the zoo is 3

Number of cages in the new zoo is 3

All animals in new zoo are ('monkey', 'camel', ('python', 'elephant', 'penguin'))

Animals brought from old zoo are ('python', 'elephant', 'penguin')

Last animal brought from old zoo is penguin

Number of animals in the new zoo is 5

## 튜플 tupdemo.py 결과

- ❑ `new_zoo = ('monkey', 'camel', ('python', 'elephant', 'penguin'))`
- ❑ `new_zoo` 의 세번째 항목 : **`new_zoo[2]`**
- ❑ `new_zoo[2]` 는 튜플이고, 이것의 세번째 항목은 **`new_zoo[2][2]`**
- ❑ 빈튜플, **`myempty = ()`**
- ❑ 항목 1개만 담는 튜플, **`singleton = ( 2, )`**
  - ❑ NOT `singleton = ( 2 )`



# 처음코딩 - Python 22

## 자료구조 - 사전

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

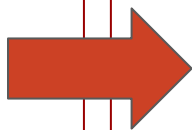
❑ 전화번호부

❑ 이름 : 전화번호

❑ 홍길동 : 999-9999

❑ 철수 : 999-5599

❑ 영희 : 999-5595



ab = {

'홍길동' : '999-9999',

'철수' : '999-5599',

'영희' : '999-5595'

}

# 사전

- ❑ 사전 = 키 + 값 으로 구성
- ❑ 키 : 정적 객체 (이름에 해당)
- ❑ 값 : 정적, 비정적 객체 모두 (주소나 연락처 등에 해당)
- ❑ `d = { key1 : value1 , key2 : value2 }`
  - ❑ \*\* 키와 값은 **콜론**으로 구분 (:)
  - ❑ \*\* 각 쌍은 **쉼표**로 구분 (,)
  - ❑ \*\* 전체는 **중괄호** { } 로 묶음

# 자료구조

## ○ 딕셔너리

딕셔너리는 하나의 요소가 **key**와 **value**로 구성되어 저장, 요소는 그 요소를 가리키는 **key**와 설명에 해당하는 **value**로 구성

```
a = {'name': 'Joy', 'phone': '010-0000-0123'}
```

딕셔너리는 { }로 묶이고, **key**와 **value** 사이에 :으로 구분됩니다.  
키(**key**)를 사용해서 값(**value**)을 읽어옵니다.



# dictdemo.py

```
# 'ab' is short for 'a'ddress'b'ook
ab = { 'Baram' : 'ludenscode@gmail.com',
       'Larry' : 'larry@wall.org',
       'Matsumoto' : 'matz@ruby-lang.org',
       'Spammer' : 'spammer@hotmail.com'
}

print("Baram's address is", ab['Baram'])

# Deleting a key-value pair
del ab['Spammer']
print('\nThere are {} contacts in the
adress-book\n'.format(len(ab)))
```

# dictdemo.py

```
for name, address in ab.items():  
    print('Contact {} at {}'.format(name, address))
```

```
# Adding a key-value pair  
ab['Gildong'] = 'gildong@ueldo.org'
```

```
if 'Gildong' in ab:  
    print("\nGildong's address is", ab['Gildong'])
```

- ❑ del 문으로 항목 삭제

  - ❑ del ab['Spammer']

- ❑ dict 의 items 메소드로 개별 '키-값' 항목에 접근

- ❑ d = { key1 : value1 , key2 : value2 }

  - ❑ \*\* 키와 값은 콜론으로 구분 (:)

  - ❑ \*\* 각 쌍은 쉼표로 구분 (,)

  - ❑ \*\* 전체는 중괄호 { } 로 묶음

# 사전

```
for name in ab:  
    print(name)
```

```
ab = {  
    'Baram' : 'ludenscode@gmail.com',  
    'Larry' : 'larry@wall.org',  
    'Matsumoto' : 'matz@ruby-lang.org',  
    'Spammer' : 'spammer@hotmail.com'  
}
```

# 사전

```
for name in ab:  
    print(name)
```

```
for name in ab:  
    print(name, ab[name])
```

```
ab = {  
    'Baram' : 'ludenscode@gmail.com',  
    'Larry' : 'larry@wall.org',  
    'Matsumoto' : 'matz@ruby-lang.org',  
    'Spammer' : 'spammer@hotmail.com'  
}
```

# 사전

```
for name in ab:  
    print(name)
```

```
for name in ab:  
    print(name, ab[name])
```

```
for name, address in ab.items():  
    print(name, address)
```

```
ab = {  
    'Baram' : 'ludenscode@gmail.com',  
    'Larry' : 'larry@wall.org',  
    'Matsumoto' : 'matz@ruby-lang.org',  
    'Spammer' : 'spammer@hotmail.com'  
}
```

```
a = {'name': 'Joy', 'phone': '010-0000-0123'}
```

```
a['name'] = 'Joy Luck'
```

```
a = {'name': 'Joy Luck', 'phone': '010-0000-0123'}
```

```
a['sex'] = 'male'
```

```
a = { 'name': 'Joy Luck',  
      'phone': '010-0000-0123',  
      'sex': 'male' }
```



처음코딩 - Python 23

# 열거형과 슬라이스연산

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com



# 열거형의 주요 기능

## 1. 멤버십 테스트

a. in / not in

b. 인덱싱 연산

i. 슬라이스 연산 - 일부분을 잘라냄

# ds\_seq1.py

```
shoplist = ['apple', 'mango', 'carrot', 'banana']  
name = 'sleep'
```

```
# Indexing or 'Subscription' operation #
```

```
print('Item 2 is', shoplist[2])  
print('Item 0 is', shoplist[0])  
print('Item 3 is', shoplist[3])  
print('Item 1 is', shoplist[1])  
print('Item -1 is', shoplist[-1])  
print('Item -2 is', shoplist[-2])  
print('Character 0 is', name[0])
```

# ds\_seq1.py 설명

```
shoplist = ['apple', 'mango', 'carrot', 'banana']  
name = 'sleep'
```

shoplist[0] 은 첫번째 항목 ('apple')

shoplist[3] 은 네번째 항목 ('banana')

shoplist[-1] 은 마지막 항목 ('banana')

shoplist[-2] 은 끝에서 두번째 항목 ('carrot')

name[0] 는 첫번째 항목 ('s')

# ds\_seq2.py

```
shoplist = ['apple', 'mango', 'carrot', 'banana']
```

```
# Slicing on a list #
```

```
print('Item 1 to 3 is', shoplist[1:3])
```

```
print('Item 2 to end is', shoplist[2:])
```

```
print('Item 1 to -1 is', shoplist[1:-1])
```

```
print('Item start to end is', shoplist[:])
```

# ds\_seq2.py 설명

```
shoplist = ['apple', 'mango', 'carrot', 'banana']
```

shoplist[1:3] 은 두번째 항목부터 3번째 항목(4번째 항목전)까지 ['mango', 'carrot']

shoplist[2:] 은 세번째 항목부터 끝까지 ['carrot', 'banana']

shoplist[1:-1] 은 두번째 항목에서 마지막 항목전까지 ['mango', 'carrot']

shoplist[:] 은 전체 항목 ['apple', 'mango', 'carrot', 'banana']

# ds\_seq2.py 설명

```
shoplist = ['apple', 'mango', 'carrot', 'banana']
```

shoplist[::1] 세번째 인수는 스텝을 의미

shoplist[::2] 는 ['apple', 'carrot']

shoplist[::-1] 는 ['banana', 'carrot', 'mango', 'apple']

– 열거형의 순서를 바꿀때 스텝에 -1 을 사용

# for comprehension

```
temperatures = [2, 5, 8, 10, 50, 80, 12, 52, 99]
# 20 이상인 수만 고르기

t_20 = []
for temperature in temperatures:
    if (temperatures>=20) t_20.append(temperatures)

print(t_20)
```

# list comprehension

```
temperatures = [2, 5, 8, 10, 50, 80, 12, 52, 99]
# 20 이상인 수만 고르기
t20 = [i for i in temperatures if i >= 20]
print(t20)
# t20 = []
# for i in temperatures:
#     if i >= 20:
#         t20.append(i)
#
# print(len(t20))
```





# 처음코딩 - Python 24

## 참조

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

객체 생성 -----> 변수에 할당

변수에 객체의 참조가 할당

```
shoplist = ['apple', 'mango', 'carrot', 'banana']  
mylist = shoplist  
  
# I purchased the first item...  
del shoplist[0]  
print('shoplist is', shoplist)  
print('mylist is', mylist)
```

```
# they point to the same object  
print('Copy by making a full slice')  
  
# Make a copy by doing a full slice  
mylist = shoplist[:]
```

```
# Remove first item  
del mylist[0]  
print('shoplist is', shoplist)  
print('mylist is', mylist)
```

- 변수 = 객체이름
  - `mylist = shoplist`
  - 주소값이 전송 / 같은 곳을 가리킴
- 변수 = 슬라이스연산자(객체)
  - `mylist = shoplist[:]`
  - 값을 복사하여 전송 / 새로운 객체 생성



# 처음코딩 - Python 12

## 함수

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

function, 함수란?

재사용 가능한 프로그램의 조각



1. 함수를 만든다.
2. 만들어진 함수에 이름을 붙인다.
3. 원하는 곳에서 함수의 이름을 부른다.

# 함수 만들어 사용

```
def say_hello():  
    print('hello world')
```

```
# 함수의 이름  
# 함수가 할 일
```

```
say_hello()
```

```
# 함수를 부른다
```

# 함수 만들기

def + 함수이름 + ( + 매개변수들 + ) + 콜론:

    함수가 할 일(명령)1

    함수가 할 일(명령)2

def say\_hello():

    print("Hello World")

# 함수, 매개변수, 인자

매개변수 : 함수로 넘겨지는 값들의 이름  
함수를 정의할 때 주어진 이름

인자 : 함수에 넘겨준 값

# 매개변수를 가진 함수의 예

```
def print_max(a, b):  
    if a > b:  
        print(a, 'is maximum')  
    elif a == b:  
        print(a, 'is equal to', b)  
    else:  
        print(b, 'is maximum')  
  
print_max(3, 4)  
  
x = 5  
y = 7  
print_max(x, y)
```

# return

```
def add(x, y):  
    return x + y
```

```
result = add(5, 3)  
print(result) # 출력: 8
```

# 함수 예제

리스트로 된 값을 읽어서 평균과 표준편차를 반환하는 함수를 만들어 보세요.

```
mydata = [10, 50, 20, 88, 90]
```

```
def data_analysis(values):
```

```
    m = 0
```

```
    sd = 0
```

```
    return (m, sd)
```

```
mean, std = data_analysis(mydata)
```

```
print(mean, std)
```

# 함수 예제

```
mydata = [10, 50, 20, 88, 90]
```

```
def data_analysis(values):  
    # 평균 구하기  
    n = len(values)  
    sum = 0  
    for i in range(n):  
        sum += values[i]  
    m = sum / n
```

```
# 표준편차  
ss = 0  
for i in range(n):  
    ss += (values[i] - m)**2  
sd = (ss / (n-1))**0.5  
  
return (m, sd)
```

```
(mean, std) = data_analysis(mydata)  
print(mean, std)
```



# 함수 예제

```
import numpy as np

def np_analysis(values):
    mean = np.mean(values)
    std = np.std(values)
    return (mean, std)

mydata = [10, 50, 20, 88, 90]
mean, std = np_analysis(mydata)
print(mean, std)
```



# 처음코딩 - Python 13

## 지역변수, 전역변수

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

변수가 의미있는 공간

- ❑ 함수안에서 변수를 만든 경우,
- ❑ 만들어진 블록 내부에서만 변수는 유효

# 지역변수 사용시 값의 변화

```
x = 50
def func(x):
    print('x 의 값은', x)
    x = 2
    print('지역변수 x 의 값은', x)

func(x)
print('x 는 여전히 ', x)
```

# 전역변수

- ❑ 상위블록에서 선언된 변수의 값을 변경할 때
- ❑ 이 변수를 전역변수로 사용한다고 알려줌
- ❑ global 사용

## 전역변수 사용 예

```
x = 50
def func():
    global x
    print('x is', x)
    x = 2
    print('Changed global x to', x)

func()
print('Value of x is', x)
```



# 처음코딩 - Python 14

## 기본인수값, 키워드인수

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com



함수를 호출할 때 사용자가 값을 넘겨주지 않으면  
자동으로 기본값을 사용하도록 할 때 사용  
이때 기본인수값은 \*상수\*

# 기본인수값 사용예

```
def say(message, times=1):  
    print(message * times)
```

```
say('Hello')
```

```
say('World', 5)
```

매개 변수의 이름을 지정하여 직접 값을 넘겨줌

## 키워드 인수 장점

- ❑ 인수 순서를 신경쓰지 않아도 된다.
- ❑ 특정한 매개변수값만 넘기고 나머지는 기본인수로 사용할 수 있다.

# 키워드 인수 장점

```
def func(a, b=5, c=10):  
    print('a is',a,'and b is',b,'and c is',c)
```

```
func(3, 7)
```

```
func(25, c=24)
```

```
func(c=50, a=100)
```

# 함수 예제

입력된 문자열을 역순으로 출력하는 `print_reverse` 함수를 정의하라.

```
[REDACTED]
```

```
print_reverse("python")  
> nohtyp
```

## 함수 예제

입력된 문자열을 역순으로 출력하는 `print_reverse` 함수를 정의하라.

```
def print_reverse(string) :  
    print(string[::-1])
```

```
print_reverse("python")
```

## 함수 예제

성적 리스트를 입력 받아 평균을 출력하는 `print_score` 함수를 정의하라.

```
[REDACTED]
```

```
[REDACTED]
```

```
print_score ([1, 2, 3])
```



## 함수 예제

성적 리스트를 입력 받아 평균을 출력하는 `print_score` 함수를 정의하라.

```
def print_score(score_list) :  
    print(sum(score_list)/len(score_list))  
  
print_score ([1, 2, 3])
```

## 함수 예제

하나의 리스트를 입력받아 짝수만 화면에 출력하는  
`print_even` 함수를 정의하라.

```
[REDACTED]
```

```
print_even ([1, 3, 2, 10, 12, 11, 15])
```

## 함수 예제

하나의 리스트를 입력받아 짝수만 화면에 출력하는  
`print_even` 함수를 정의하라.

```
def print_even (my_list) :  
    for v in my_list :  
        if v % 2 == 0 :  
            print(v)
```

```
print_even ([1, 3, 2, 10, 12, 11, 15])
```

## 함수 예제

하나의 리스트를 입력받아 짝수만 화면에 출력하는  
`print_even` 함수를 정의하라.

```
def print_even (my_list) :  
    rst = [v for v in my_list if v % 2 == 0]  
    print(rst)
```

```
print_even ([1, 3, 2, 10, 12, 11, 15])
```

## 함수 예제

문자열 하나를 입력받아 인터넷 주소를 반환하는  
`make_url` 함수를 정의하라.

```
def make_url(word):
```

```
    url = "http://" + word + ".com"
```

```
    return url
```

```
make_url("naver")
```

# 함수 예제

문자열 하나를 입력받아 인터넷 주소를 반환하는  
make\_url 함수를 정의하라.

```
def make_url(string) :  
    url = "www." + string + ".com"  
    return url
```

```
make_url("naver")
```

# 함수 예제

문자열을 입력받아 각 문자들로 구성된 리스트로 반환하는 `make_list` 함수를 정의하라.



`make_list("abcd")`  `['a', 'b', 'c', 'd']`

## 함수 예제

문자열을 입력받아 각 문자들로 구성된 리스트로 반환하는 `make_list` 함수를 정의하라.

```
def make_list (string) :  
    my_list = []  
    for i in string :  
        my_list.append(i)  
    return my_list
```

`make_list("abcd")`  `['a', 'b', 'c', 'd']`





처음코딩 - Python 15

# VarArgs 매개변수

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

임의 개수의 매개 변수 지정시 사용  
별(\*) 기호 사용하여 표현

# 사용법

튜플형 변수들 : (2,3,4)

```
def total(initial=5, *numbers, **keywords) :
```

딕셔너리형 변수들 : a=1, b=2, c=4

# 사용예

```
def total(initial=5, *numbers, **keywords):  
    count = initial  
    for number in numbers:  
        count += number  
    for key in keywords:  
        count += keywords[key]  
    return count  
  
print(total(10, 1, 2, 3, vegetables=50, fruits=100))
```



# 처음코딩 - Python 16

## return

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

return

함수의 명령이 실행된 후 결과를 돌려줄 때 사용

# 사용예

```
def maximum(x, y):  
    if x > y:  
        return x  
    elif x == y:  
        return 'The numbers are equal'  
    else:  
        return y  
  
print(maximum(2, 3))
```

# return None

`return` 문을 지정하지 않으면 `return None` 실행

`None` 은 변수에 할당할 값이 없다는 의미



# None & pass

```
def some_function():  
    pass  
  
print(some_function())
```

프로그램에 대한 설명서를 작성  
함수에 포함된 첫 논리적 명령행의 문자열

# DocString, 설명문자열

```
def print_max(x, y):  
    '''Prints the maximum of two numbers.  
  
    The two values must be integers.'''  
    x = int(x)  
    y = int(y)  
    if x > y:  
        print(x, 'is maximum')  
    else:  
        print(y, 'is maximum')  
  
print_max(3, 5)  
print(print_max.__doc__)  
help(print_max)
```



# 처음코딩 - Python 17

## 모듈이해

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

여러 함수를 한꺼번에 불러들여 재사용

- ❑ `.py` 확장자를 가진 파일안에 함수와 변수를 정의
- ❑ 파이썬 인터프리터를 만든 언어로 모듈 작성 (ex. C언어)

# 모듈 사용법

□ import 명령으로 모듈을 불러와 사용

```
import sys
```

```
...
```

```
for i in sys.argv:
```

```
...
```

sys 라는 이름을 가진 모듈에서

argv 라는 이름을 가진 변수 사용

```
import sys
```

```
print('The command line arguments are:')
```

```
print(sys.argv)
```

```
print('\n\nPATH is')
```

```
print(sys.path)
```

**module\_s1.py 로 저장**

**> python module\_s1.py we play**



```
import sys

print('The command line arguments are:')
for i in sys.argv:
    print(i)

print('\n\nPATH is')
for i in sys.path:
    print(i)
```

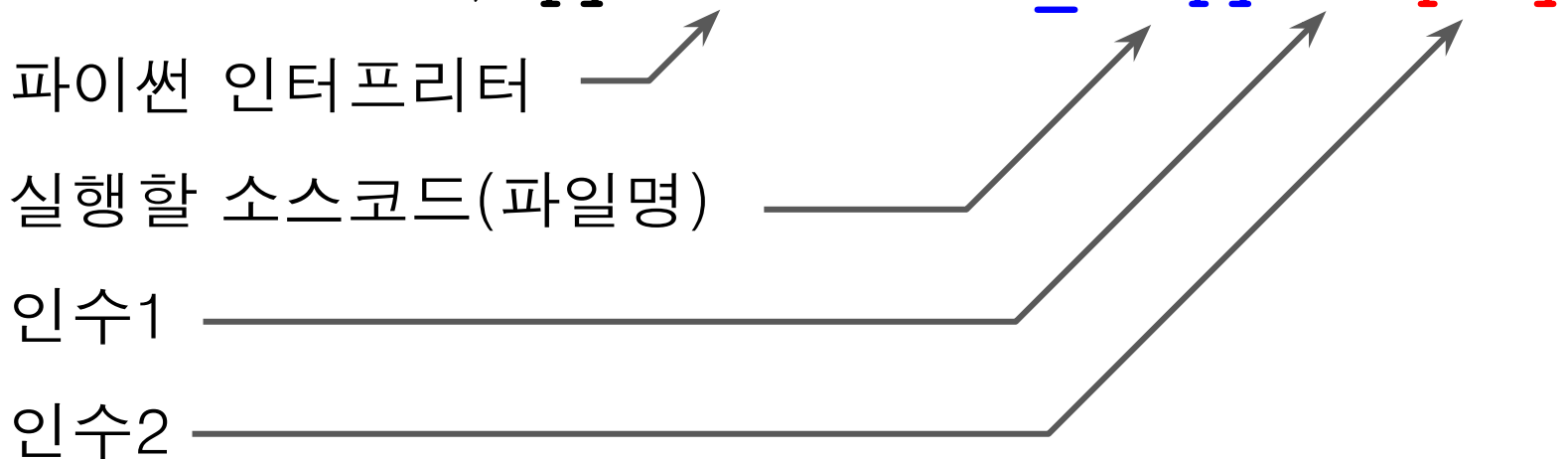
**module\_s2.py 로 저장**

**> python module\_s2.py we play**

# sys.argv 보충 설명


- ❑ 문자열의 리스트
- ❑ 명령줄 인수들의 리스트

ex) **python** **module\_s1.py** **we** **play**



# sys.argv 저장된 값

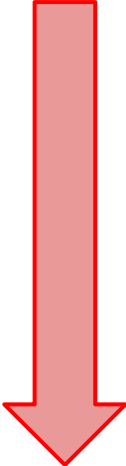
ex) `python module_s1.py we play`

파이썬 인터프리터 

실행할 소스코드(파일명) 

인수1 

인수2 



```
sys.argv = ["module_s1.py" , "we" , "play"]
```

# sys.argv 리스트 인덱스

- ❑ `sys.argv = ["module_s1.py" , "we" , "play"]`
- ❑ **`sys.argv[0]`** <<-- **`"module_s1.py"`**
- ❑ `sys.argv[1]` <<-- `"we"`
- ❑ `sys.argv[2]` <<-- `"play"`

# sys.path 리스트 인덱스

```
❏ sys.path = [ "", "C:\Pyth...", ... ]
```

```
❏ sys.path[0] <|-- ""
```

```
❏ sys.argv[1] <|-- "C:\Python\Python36"
```

```
❏ sys.argv[2] <|-- "C:\Python\Python36\DLLs"
```

```
❏ ...
```

# 바이트 컴파일된 .pyc

- ❑ .pyc 는 바이트 컴파일된 중간 단계의 파일
  - ❑ .pyc 는 .py 파일이 저장된 곳에 생성
  - ❑ 선행작업을 통해 .pyc 생성
  - ❑ .pyc 는 속도향상을 가져옴

## from ... import 문

- ❑ `sys.argv` 대신 `argv` 사용할 때
  - ❑ `from sys import argv` 구문 사용

```
from math import sqrt  
print("Square root of 16 is", sqrt(16))
```



# 처음코딩 - Python 18

## 모듈 만들기

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com



## 모듈의 이름(name) 속성

- ❑ 모든 모듈은 이름을 가지고 있음
- ❑ 모듈내 포함된 명령으로 모듈의 이름 확인
  - ❑ 실행되었을 때 : `__main__`
  - ❑ `import` 되었을 때 : 모듈의 이름

## 모듈의 name 확인

```
if __name__ == '__main__':  
    print('I am running. My name is ', end='')  
    print(__name__)  
  
else:  
    print('I'm imported. My name is ', end='')  
    print(__name__)
```

```
if __name__ == '__main__':  
    print('I am running. My name is ', end='')  
    print(__name__)
```

**> python module\_name.py**

```
else:
```

```
    print('I'm imported. My name is ', end='')  
    print(__name__)
```

**> python**

**>>> import module\_name**

.py 확장자를 가진 이름으로 저장

```
def say_hi():  
    print('Hi, Mymodule speaking.')
```

```
__version__ = '0.1'
```

mymodule.py 로 저장

## mymodule.py 내부의 함수와 변수 사용

```
import mymodule  
mymodule.say_hi()  
  
print('Version', mymodule.__version__)
```

mymoduledemo1.py 로 저장

from ... import 를 사용

```
from mymodule import say_hi, __version__
```

```
say_hi()
```

```
print('Version', __version__)
```

mymoduledemo2.py 로 저장

```
from ... import *
```

```
from mymodule import *
```

```
say_hi()
```

```
# import * 형식은 가급적 피할것
```

```
# 이경우 __를 사용한 변수는 불러오지 않음
```

mymoduledemo3.py 로 저장



# Zen of Python (파이썬정신)

```
import this
```

Beautiful is better than ugly.  
Explicit is better than implicit.  
Simple is better than complex.  
Complex is better than complicated.  
Flat is better than nested.  
Sparse is better than dense.  
Readability counts.  
Special cases aren't special enough to break the rules.  
Although practicality beats purity.  
Errors should never pass silently.  
Unless explicitly silenced.  
In the face of ambiguity, refuse the temptation to guess.  
There should be one-- and preferably only one --obvious way to do it.  
Although that way may not be obvious at first unless you're Dutch.  
Now is better than never.  
Although never is often better than \*right\* now.  
If the implementation is hard to explain, it's a bad idea.  
If the implementation is easy to explain, it may be a good idea.  
Namespaces are one honking great idea -- let's do more of those!



# 처음코딩 - Python 19

## dir 내장함수

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

## 객체(모듈)에 정의된 식별자(이름) 불러오기

- ❑ 모듈은 함수, 클래스, 변수를 가지고 있음
- ❑ `dir( )` 은 모듈에 포함된 식별자 목록을 반환

```
$ python
```

```
>>> import sys
```

```
# sys 모듈 내에 선언된 속성들의 식별자 이름 목록
```

```
>>> dir(sys)
```

```
['__displayhook__', '__doc__',  
'argv', 'builtin_module_names',  
'version', 'version_info']
```

```
# 실제로는 너무 길어 여기에는 몇 개만 적었음
```

```
> python
```

```
>>> import sys
```

```
>>> for num in dir(sys):
```

```
...     print(num)
```

```
...
```

```
__displayhook__
```

```
__doc__
```

```
__excepthook__
```

```
__interactivehook__
```

```
__loader__
```

```
__name__
```

```
$ python
```

```
>>> import sys
```

```
# 현재 모듈에 선언된 속성들의 식별자 이름 목록
```

```
>>> dir()
```

```
['__builtins__', '__doc__',  
 '__name__', '__package__']
```

```
$ python
```

```
>>> import sys
```

```
## 새로운 변수 'a' 생성
```

```
>>> a = 5
```

```
>>> dir()
```

```
['__builtins__', '__doc__', '__name__',  
'__package__', 'a']
```



```
$ python
```

```
>>> import sys
```

```
# 식별자 이름 제거
```

```
>>> del a
```

```
>>> dir()
```

```
['__builtins__', '__doc__', '__name__',  
 '__package__']
```



# 처음코딩 - Python 25

## 문자열 보충

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

파이썬의 모든 문자열은 str 클래스의 객체

# help(str)

```
name = 'Baram'
if name.startswith('Bar'):
    print('Yes, the string starts with "Bar"')

if 'a' in name:
    print('Yes, it contains the string "a"')

if name.find('ara') != -1:
    print('Yes, it contains the string "ara"')
```

```
delimiter = '_*_  
mylist = ['Brazil', 'Russia', 'India', 'China']  
print(delimiter.join(mylist))
```

startswith()

in

find()

join()



# 처음코딩 - Python 26

## 백업 프로그램 1

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

파이썬으로 만들어보는 백업프로그램



- ❑ 커맨드 명령으로 사용되며 zip 과 unzip 명령을 이용한다.
- ❑ <http://stahlworks.com/dev/?tool=zipunzip>
- ❑ zip & unzip 을 다운로드 후 C:\utils 를 만들고 내부에 복사
- ❑ path 설정
  - ❑ 내 PC >> 속성 >> (시스템 >> 정보) >> 고급 시스템 설정 >> 환경 변수 >> 시스템 변수 >> Path - 편집 >> 새로 만들기 >> C:\utils - 확인
  - ❑ cmd (명령프롬프트) 에서 zip 명령 실행 확인

- ❑ 백업할 파일과 디렉토리들은 리스트의 형태로 지정해 둔다.
- ❑ 주 백업 디렉토리를 두고, 백업은 그 안에 저장되어야 한다.
- ❑ 백업된 파일들은 zip파일로 압축해 둔다.
- ❑ zip 파일의 이름은 현재 날짜와 시간으로 한다.
- ❑ 커맨드 명령으로 사용되며 설치한 zip 명령을 이용한다.  
(참고: 명령줄 인터페이스에서 사용할 수 있는 어떤 압축 유틸리티든지 사용이 가능)

# 프로그램 소스

```
import os
import time

# zip 파일 다운로드시

# 1. The files and directories to be backed up are
# specified in a list.
# Example on Windows:
# source = ['"C:\\My Documents"', 'C:\\Code']
# Example on Mac OS X and Linux:
source = ['D:\\temp']
# Notice we had to use double quotes inside the string
# for names with spaces in it.
```

# 프로그램 소스

```
# 2. The backup must be stored in a
# main backup directory
# Example on Windows:
# target_dir = 'E:\\Backup'
# Example on Mac OS X and Linux:
target_dir = 'D:\\backup'
# Remember to change this to which folder you will be using

# 3. The files are backed up into a zip file.
# 4. The name of the zip archive is the current date and time
target = target_dir + os.sep + \
        time.strftime('%Y%m%d%H%M%S') + '.zip'
```

# 프로그램 소스

```
# Create target directory if it is not present
if not os.path.exists(target_dir):
    os.mkdir(target_dir)    # make directory

# 5. We use the zip command to put the files in a zip archive
zip_command = "zip -r {0} {1}".format(target, ' '.join(source))
# Run the backup
print("Zip command is:")
print(zip_command)
print("Running:")
if os.system(zip_command) == 0:
    print('Successful backup to', target)
else:
    print('Backup FAILED')
```

# SW 개발단계

1. 무엇을 만들 것인가? (분석 단계)
2. 어떻게 만들 것인가? (설계 단계)
3. 만들기 (구현 단계)
4. 테스트 하기 (테스트와 디버깅 단계)
5. 실제로 사용하기 (활용 또는 배포 단계)
6. 유지 및 보수하기 (개선 단계)

# backup\_ver2.py

```
import os
import time

# 1. The files and directories to be backed up are
# specified in a list.
# Example on Windows:
# source = ['"C:\\My Documents"', 'C:\\Code']
# Example on Mac OS X and Linux:
source = ['/Users/swa/notes']
# Notice we had to use double quotes inside the string
# for names with spaces in it.
```

# backup\_ver2.py

```
# 2. The backup must be stored in a
# main backup directory
# Example on Windows:
# target_dir = 'E:\\Backup'
# Example on Mac OS X and Linux:
target_dir = '/Users/swa/backup'
# Remember to change this to which folder you will be using

# Create target directory if it is not present
if not os.path.exists(target_dir):
    os.mkdir(target_dir) # make directory
```



# backup\_ver2.py

```
# 3. The files are backed up into a zip file.
# 4. The current day is the name of the subdirectory
# in the main directory.
today = target_dir + os.sep + time.strftime('%Y%m%d')
# The current time is the name of the zip archive.
now = time.strftime('%H%M%S')
# The name of the zip file
target = today + os.sep + now + '.zip'

# Create the subdirectory if it isn't already there
if not os.path.exists(today):
    os.mkdir(today)
    print 'Successfully created directory', today
```

# backup\_ver2.py

```
# 5. We use the zip command to put the files in a zip archive
zip_command = "zip -r {0} {1}".format(target,
                                     ' '.join(source))

# Run the backup
print "Zip command is:"
print zip_command
print "Running:"
if os.system(zip_command) == 0:
    print 'Successful backup to', target
else:
    print 'Backup FAILED'
```

변경된 부분 - 주 백업 디렉토리 안에 그 날짜에 해당하는 디렉토리가 있는지 여부를 `os.path.exists` 함수로 확인

`os.mkdir` 함수를 통해 주 백업 디렉토리 안에 그 날짜에 해당하는 디렉토리가 있는지 여부를 `os.path.exists` 함수로 확인, 해당 디렉토리가 없으면, `os.mkdir` 함수를 통해 디렉토리 생성

# backup\_ver3.py

```
import os
import time

# 1. The files and directories to be backed up are
# specified in a list.
# Example on Windows:
# source = ['"C:\\My Documents"', 'C:\\Code']
# Example on Mac OS X and Linux:
source = ['/Users/swa/notes']
```

```
# Notice we had to use double quotes inside the
string
# for names with spaces in it.

# 2. The backup must be stored in a
# main backup directory
# Example on Windows:
# target_dir = 'E:\\Backup'
# Example on Mac OS X and Linux:
```

## backup\_ver3.py

```
target_dir = '/Users/swa/backup'
# Remember to change this to which folder you will
be using

# Create target directory if it is not present
if not os.path.exists(target_dir):
    os.mkdir(target_dir) # make directory

# 3. The files are backed up into a zip file.
```

```
# 4. The current day is the name of the
subdirectory
# in the main directory.
today = target_dir + os.sep +
time.strftime('%Y%m%d')
# The current time is the name of the zip archive.
now = time.strftime('%H%M%S')
```

# backup\_ver3.py

```
# Take a comment from the user to
# create the name of the zip file
comment = raw_input('Enter a comment --> ')
# Check if a comment was entered
if len(comment) == 0:
    target = today + os.sep + now + '.zip'
else:
    target = today + os.sep + now + '_' +
        comment.replace(' ', '_') + '.zip'
```



```
# Create the subdirectory if it isn't already there
if not os.path.exists(today):
    os.mkdir(today)
    print 'Successfully created directory', today
```

```
# 5. We use the zip command to put the files in a
zip archive
```

# backup\_ver3.py

```
zip_command = "zip -r {0} {1}".format(target,  
                                     '  
'  
                                     '.join(source))  
  
# Run the backup  
print "Zip command is:"  
print zip_command  
print "Running:"  
if os.system(zip_command) == 0:
```

```
    print 'Successful backup to', target  
else:  
    print 'Backup FAILED'
```

동작하지 않는 프로그램  
버그를 찾아라

# backup\_ver4.py

```
import os
import time

# 1. The files and directories to be backed up are
# specified in a list.
# Example on Windows:
# source = ['"C:\\My Documents"', 'C:\\Code']
# Example on Mac OS X and Linux:
source = ['/Users/swa/notes']
# Notice we had to use double quotes inside the string
# for names with spaces in it.
```

# backup\_ver4.py

```
# 2. The backup must be stored in a  
# main backup directory  
# Example on Windows:  
# target_dir = 'E:\\Backup'  
# Example on Mac OS X and Linux:  
target_dir = '/Users/swa/backup'  
# Remember to change this to which folder you will  
be using
```

```
# Create target directory if it is not present
if not os.path.exists(target_dir):
    os.mkdir(target_dir) # make directory

# 3. The files are backed up into a zip file.
```

# backup\_ver4.py

```
# 4. The current day is the name of the subdirectory
# in the main directory.
today = target_dir + os.sep + time.strftime('%Y%m%d')
# The current time is the name of the zip archive.
now = time.strftime('%H%M%S')

# Take a comment from the user to
# create the name of the zip file
comment = raw_input('Enter a comment --> ')
# Check if a comment was entered
```



# backup\_ver4.py

```
# Check if a comment was entered
if len(comment) == 0:
    target = today + os.sep + now + '.zip'
else:
    target = today + os.sep + now + '_' + \
        comment.replace(' ', '_') + '.zip'

# Create the subdirectory if it isn't already there
if not os.path.exists(today):
    os.mkdir(today)
    print 'Successfully created directory', today
```

# backup\_ver4.py

```
# 5. We use the zip command to put the files in a zip archive
zip_command = "zip -r {0} {1}".format(target,
                                     ' '.join(source))

# Run the backup
print "Zip command is:"
print zip_command
print "Running:"
if os.system(zip_command) == 0:
    print 'Successful backup to', target
else:
    print 'Backup FAILED'
```

더 필요한 개선사항은?

`os.system`

VS

`zipfile , tarfile`

소프트웨어는 성장하는 것이지,  
만들어지는 것이 아니다.

(Great software is not built, it is grown)

- Bill de h0ra



# 처음코딩 - Python 27

## 객체지향

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

## ❑ 절차지향 프로그래밍

- ❑ 함수 조합

## ❑ 객체지향 프로그래밍

- ❑ 데이터와 기능을 객체로 묶어서 구성

# 객체지향에서 클래스와 객체

- 클래스

- 새로운 형식, 틀

- 객체

- 클래스의 인스턴스



# 객체지향에서 클래스의 속성

- ❑ 클래스에 속한 변수
  - ❑ 필드(field)
- ❑ 클래스에 속한 함수
  - ❑ 메소드(method)

# 필드의 종류

- ❑ 클래스의 인스턴스/객체에 내장된 변수
  - ❑ 인스턴스 변수
- ❑ 클래스 자체에 내장된 변수
  - ❑ 클래스 변수

- 클래스 메소드는 일반적인 함수와 딱 한 가지 다른 점이 있는데, 그것은 메소드의 경우 매개 변수의 목록에 항상 추가로 한 개의 변수가 맨 앞에 추가되어야 한다는 점입니다. 또한 메소드를 호출할 때 이 변수에는 우리가 직접 값을 넘겨주지 않으며, 대신 파이썬이 자동으로 값을 할당합니다. 이 변수에는 현재 객체 자신의 참조가 할당되며, 일반적으로 **self** 라 이름을 짓습니다.

# 단순한 클래스 예시

```
class Person:  
    pass # An empty block  
  
p = Person()  
  
print(p)
```

# 동작원리

1. 생성
2. 이름
3. 블록
4. 객체 생성
5. 결과 확인

메소드는 (self 를 제외하고)  
함수와 동일

```
class Person:  
    def say_hi(self):  
        print('Hello, how are you?')
```

```
p = Person()  
p.say_hi()
```

self 는 아무 매개변수를 넘겨받지 않지만 함수정의에  
self 를 가짐

- ❑ `init` 메소드는 클래스가 인스턴스화 될 때 호출
- ❑ 객체가 생성시 초기화 명령이 필요할 때 유용하게 사용
- ❑ `init`의 앞과 뒤에 있는 밑줄은 두 번씩 입력



# init 메소드

```
class Person:
    def __init__(self, name):
        self.name = name
    def say_hi(self):
        print('Hello, my name is', self.name)

p = Person('Baram')
p.say_hi()
```

# 클래스 변수와 객체 변수

- ❑ 클래스변수

- ❑ 공유됨

- ❑ 모든 인스턴스들이 접근할 수 있음

- ❑ 한개만 존재

- ❑ 객체가 값을 변경하면 다른 인스턴스에 적용됨

# 클래스 변수와 객체 변수

- ❑ 객체 변수

- ❑ 개별 객체 / 인스턴스에 속함 (독립)

- ❑ 다른 인스턴스들이 접근할 수 없음

```
class Robot:
```

```
    """Represents a robot, with a name."""
```

```
    # A class variable, counting the number of robots  
    population = 0
```

```
    def __init__(self, name):
```

```
        """Initializes the data."""
```

```
        self.name = name
```

```
        print("(Initializing {})".format(self.name))
```

```
# When this person is created, the robot  
# adds to the population  
Robot.population += 1
```

```
def die(self):  
    """I am dying."""  
    print("{} is being destroyed!".format(self.name))  
  
    Robot.population -= 1
```

```
if Robot.population == 0:  
    print("{} was the last one.".format(self.name))
```

```
else:  
    print("There are still {:d} robots \\  
        working.".format(Robot.population))
```

```
def say_hi(self):  
    """Greeting by the robot.
```

```
Yeah, they can do that."""
```

```
print("Greetings, my masters call me \
      {}".format(self.name))
```

```
@classmethod
```

```
def how_many(cls):
```

```
    """Prints the current population."""
```

```
    print("We have {:d} robots.".format(cls.population))
```

```
droid1 = Robot("R2-D2")
droid1.say_hi()
Robot.how_many()
droid2 = Robot("C-3PO")
droid2.say_hi()
Robot.how_many()

print("\nRobots can do some work here.\n")
print("Robots have finished work. So let's destroy them.")
```



```
droid1.die()
```

```
droid2.die()
```

```
Robot.how_many()
```

`@classmethod`

클래스메소드로 만들어 줌

```
@classmethod
```

```
def how_many(cls):
```

```
how_many = classmethod(how_many)
```



# 처음코딩 - Python 28

## 상속

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

객체지향의 장점

코드재활용

상속

클래스간의  
크고 작은 형식을 구현

# 교수와 학생들의 명부 작성 프로그램

교수 & 학생 : 이름, 나이, 주소

교수 만 : 연봉, 과목, 휴가

학생 만 : 성적, 등록금

## 방법1 : 2개의 클래스

교수 - 이름, 나이, 주소, 연봉, 과목, 휴가

학생 - 이름, 나이, 주소, 성적, 등록금



## 방법2 : 공통 클래스와 상속

공통클래스(SchoolMember) - 이름, 나이, 주소

교수(Teacher) - 연봉, 과목, 휴가 + 공통클래스

학생(Student) - 성적, 등록금 + 공통클래스

```
class SchoolMember:
    '''Represents any school member.'''
    def __init__(self, name, age):
        self.name = name
        self.age = age
        print('(Initialized SchoolMember: \
              {})' .format(self.name))

    def tell(self):
        '''Tell my details.'''
        print('Name:"{}" Age:"{}"'.format(self.name, \
```

```
        self.age), end=' ')
```

```
class Teacher(SchoolMember):
```

```
    '''Represents a teacher.'''
```

```
    def __init__(self, name, age, salary):
```

```
        SchoolMember.__init__(self, name, age)
```

```
        self.salary = salary
```

```
        print('(Initialized Teacher: {})'.format(self.name))
```

```
    def tell(self):
```

```
        SchoolMember.tell(self)
```

```
print('Salary: "{:d}"'.format(self.salary))
```

```
class Student(SchoolMember):
```

```
    '''Represents a student.'''
```

```
    def __init__(self, name, age, marks):
```

```
        SchoolMember.__init__(self, name, age)
```

```
        self.marks = marks
```

```
        print('(Initialized Student: {})'.format(self.name))
```

```
    def tell(self):
```

```
        SchoolMember.tell(self)
```

```
print('Marks: "{:d}"'.format(self.marks))
```

```
t = Teacher('Mrs. Shrividya', 40, 30000)
```

```
s = Student('Swaroop', 25, 75)
```

```
# prints a blank line
```

```
print()
```

```
members = [t, s]
```

```
for member in members:
```

```
    # Works for both Teachers and Students
```

```
    member.tell()
```



# 처음코딩 - Python 29

## 입력과 출력

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

palindrome

```
def reverse(text):  
    return text[::-1]  
  
def is_palindrome(text):  
    return text == reverse(text)  
  
something = input("Enter text: ")  
if is_palindrome(something):  
    print("Yes, it is a palindrome")  
else:  
    print("No, it is not a palindrome")
```



- ❑ 슬라이스 사용 : 문자열 뒤집기
- ❑ `palindrome`
  - ❑ 뒤집힌 문자열과 원래의 문자열이 일치하는 것

# paldrome 프로그램 개선

- ❑ 문장부호, 공백 등의 포함한 문자열의 paldrome 의 검사할 수 있는 프로그램으로 개선해 보세요.
- ❑ ex. “Rise to vote, sir.”

# 파일 입출력

1. file 클래스의 객체 생성
2. 파일을 연다.
3. 파일에 내용을 쓰거나 읽는다.
  - a. read, readline, write 등의 메소드 활용
4. 파일을 닫는다.
  - a. close 메소드로 종료

# 파일 입출력

```
poem = '''\
Programming is fun
When the work is done
if you wanna make your work also fun:
    use Python!
'''

# Open for 'w'riting
f = open('poem.txt', 'w')
```

```
# Write text to file
```

```
f.write(poem)
```

```
# Close the file
```

```
f.close()
```

```
# If no mode is specified,
```

```
# 'r'ead mode is assumed by default
```

```
f = open('poem.txt')
```

```
while True:
    line = f.readline()
    # Zero length indicates EOF
    if len(line) == 0:
        break
    # The `line` already has a newline
    print(line, end="")
# close the file
f.close()
```

# Pickle

파이썬 객체를 파일로 저장  
객체를 "영구히" 저장

# pickle

pickle 모듈을 가져온다.

```
>> import pickle
```

파일을 열때 b (binary) 모드를 사용한다.

```
>> f = open(shoplistfile, 'wb')
```

```
>> f = open(shoplistfile, 'rb')
```

pickle 의 dump() 함수를 사용한다.

```
>> pickle.dump(shoplist, f)
```



# pickle

```
import pickle

# The name of the file where we will store the object
shoplistfile = 'shoplist.data'
# The list of things to buy
shoplist = ['apple', 'mango', 'carrot']
# Write to the file
f = open(shoplistfile, 'wb')
# Dump the object to a file
pickle.dump(shoplist, f)
f.close()
```

# pickle

```
# Destroy the shoplist variable
del shoplist

# Read back from the storage
f = open(shoplistfile, 'rb')
# Load the object from the file
storedlist = pickle.load(f)
print(storedlist)
```



# 처음코딩 - Python 30

## 예외처리

바람의 동영상강의 유튜브 채널 <https://www.youtube.com/user/EVENTIA77>

email : ludenscode@gmail.com

비상사태 발생시

제대로 대처하는 것이 중요 !!!

# 에러발생시

```
>>> f = open("없는파일.zip", 'r')
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
FileNotFoundError: [Errno 2] No such file or directory: '없는파일.zip'
```

```
>>> 4/0
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
ZeroDivisionError: division by zero
```

```
>>> a=[0,1,2]
```

```
>>> a[3]
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
IndexError: list index out of range
```

# 에러발생시

에러는 try..except 문을 통해 처리

```
try:
```

```
...
```

```
except:
```

```
...
```

오류발생시 except 블록 수

# exceptions\_handle.py

```
try:
    text = input('Enter something --> ')
except EOFError:
    print("Why did not do an EOF on me?")
except KeyboardInterrupt:
    print("You cancelled the operation.")
else:
    print("You entered {}".format(text))
```

# python 참고자료

- ❖ 위키독스, 점프 투 파이썬 - <https://wikidocs.net/>
- ❖ A Byte of Python - <https://goo.gl/HEfFqd>
- ❖ CodeCademy - <https://goo.gl/hq3dMN>
- ❖ 처음코딩 - <https://opentutorials.org/course/2967>
- ❖ 강의영상 - <https://www.youtube.com/user/EVENTIA77>
- ❖ 소스코드 - [https://github.com/eventia/ZBC\\_Python](https://github.com/eventia/ZBC_Python)



**END**