



클라우드 컴퓨팅과 AI서비스 (3주차)

융합학과 권오영
oykwon@koreatech.ac.kr

학습내용

- ❖ 파이썬 개발 환경
- ❖ 파이썬 언어 소개

Computational Thinking: A beginner's guide to problem-solving and programming

written by Karl Beecher

컴퓨팅사고 정리

컴퓨팅사고란? (What is computational thinking?)

- ❖ 컴퓨터를 활용하여 문제를 해결하는 방안
- ❖ 유사한 개념인 Procedural Thinking (Papert, 1980): 컴퓨터를 사용해서 문제를 해결하는 방법으로 주어진 문제에 대하여 컴퓨터가 수행할 수 있는 알고리즘적인 해결책을 찾는 방안
- ❖ CT의 핵심개념(core concept)
 - logical thinking, algorithmic thinking, decomposition, generalisation and pattern recognition, modelling, abstraction, evaluation
- ❖ CT의 주변개념(peripheral concept)
 - data representation, critical thinking, computer science, automation, simulation/visualisation

컴퓨팅사고란? (What is computational thinking?)

❖ 컴퓨팅사고의 사용 예

- Pipelining a graduation ceremony
- Predicting climate change
- Sorting music charts (Sorting exam score)
- Assisting police, lawyers and judges

❖ 컴퓨팅사고 ≠ Computer Science

- 컴퓨팅사고는 문제해결의 하나의 접근법으로 컴퓨터 과학의 원리와 사례들을 활용하여 컴퓨터에서 수행 가능한 해결책을 만드는 것이다. 단지 프로그래머를 위한 것이 아니다. 컴퓨팅 사고는 다양한 영역에 적용 가능하다.

컴퓨터 소프트웨어

컴퓨터 소프트웨어

- ❖ Computer Software is a set of program instructions, including related data and documentation, that can be executed by computer.
- ❖ We need an artificial language to write a program.
- ❖ Each programming language has a set of primitive constructs, a syntax, a static semantics, and a semantics.
 - Primitive constructs ~ words
 - Syntax ~ 문장을 구성하는 방법 (문법) : cat dog boy.
 - Static semantics ~ 문장이 의미가 있는가를 정의:
I are big. (are -> am)
 - Semantics ~ 문장이 가지고 있는 진정한 의미

컴퓨터 소프트웨어

❖ Python

- Primitive constructs ~ literals (number, string),
infix operators (+, /), etc.
- Syntax ~ $3.2 + 3.2$
- Static semantics ~ $3.2/abc$
(문법적으로는 맞지만 숫자를 문자로 나눌 수 없어 static semantics를 위반)
- Semantics ~ 올바른 프로그램은 한가지 의미만을 가짐
(자연어는 모호함이 있을 수 있다. 즉 동일한 문장이지만 상황에 따라 칭찬일 수도 있고 비난일 수도 있다.)

컴퓨터 소프트웨어

❖ Syntax error

- 가장 흔하지만 그리 심각한 문제를 야기하지는 않음
- 대부분의 언어는 문법오류가 발생한 지점을 잘 알려준다.

❖ Static semantic error

- 배열의 인덱스 범위, 서로 다른 타입의 연산 등
- Java: static semantic checking 을 많이 실행
- C, Python: static semantic checking이 약함

❖ Semantic error: 심각한 오류 (프로그램 논리의 오류)

컴퓨터 소프트웨어

- ❖ Semantic error: 심각한 오류 (프로그램 논리의 오류)
 - Crash (메모리 참조 오류등으로 프로그램이 중단됨)
 - Never stop (종결조건의 오류등으로 무한루프에 빠지는 상황)
 - ✓ 프로그램 수행시 중간 결과값들을 출력하게 코딩 (개발시)
 - 수행을 마치고 결과를 생성
 - ✓ 결과가 올바를 수 도 있고, 틀릴 수 도 있음
- ❖ 디버깅(Debugging): 오류를 발견하고 수정하는 과정
- ❖ Software Testing: 소프트웨어가 기대하는 결과와 일치하는 실제 결과를 내놓지 점검하는 과정
 - 소프트웨어 개발 모든 과정에서 테스트는 필요
(TDD; Test Driven Development)
 - 결과가 불일치할 경우 디버깅 수행

컴퓨터 소프트웨어

❖ Programming Step (Problem solving step)

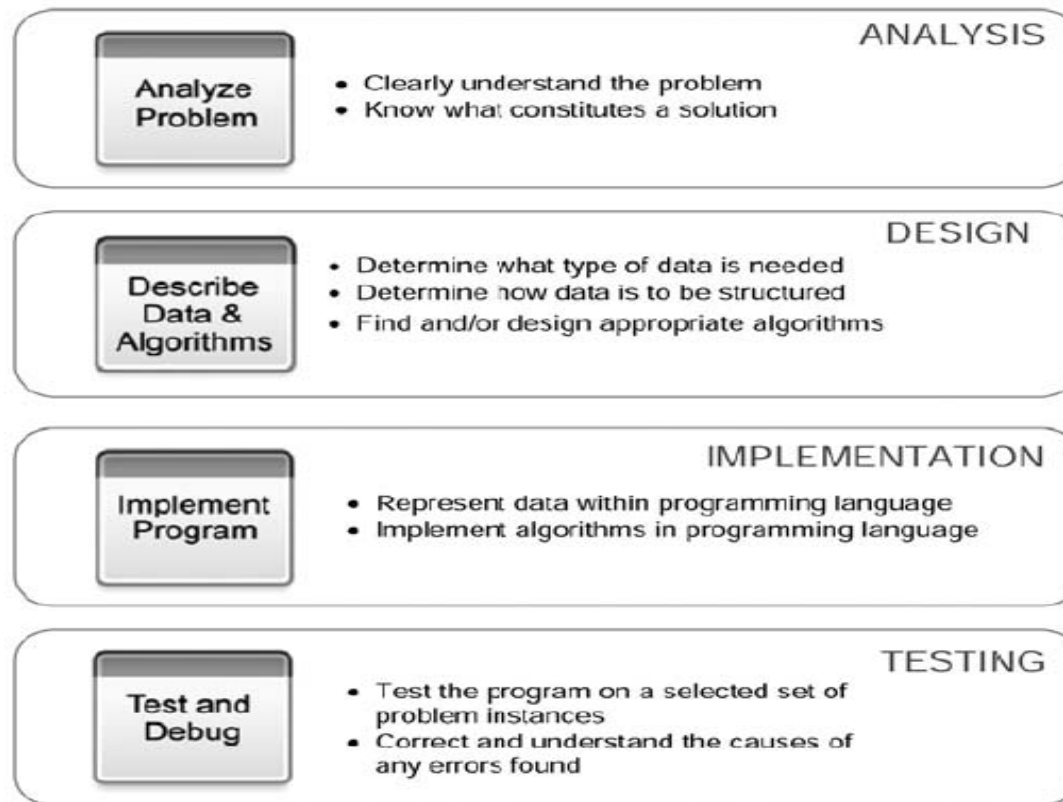


FIGURE 1-22 Process of Computational Problem Solving

디버깅 힌트

- ❖ print -> 가장 단순하고, 가장 중요한 디버깅 도구
 - 의심나는 곳 전후에 프린트 문을 삽입

- ❖ *Look for the usual suspects.* E.g., have you (일반적인 용의자를 찾아라. 당신은)
 - 잘못된 순서로 함수에 인수를 전달했는가?
 - 이름을 잘못 입력했는가? (대문자대신 소문자를 입력)
 - 변수를 다시 초기화하는데 실패했는가?
 - 두 실수(부동소수점)의 값을 비교할 때 거의 같음 대신 같음으로 비교했는가?
(실수연산은 학교에서 배웠던 연산과 동일하지 않다. -> 왜?)
 - 두 객체의 동일성을 비교했을 때 값의 동일성을 비교했는가?
(값이 같음을 비교할 수 도 있고, 구조가 같음을 비교할 수 도 있다. 상황에 따라 올바른 비교를 했는지 검증할 필요가 있다.)
 - 일부 내장(built-in) 함수가 부작용을 일으킨다는 것을 잊었는가?
 - 함수를 호출할 때 함수 이름만 쓰고 ()를 잊었는가?
 - 의도하지 않은 별칭을 만들었는가?
 - 당신이 일반적으로 만드는 다른 실수를 하지 않았는가?

디버깅 힌트

- ❖ 왜 프로그램이 원하는 작업을 수행하지 않는지 묻지 말고, 대신 왜 그것이 무엇을 하고 있는지 물어보자. (*Stop asking yourself why the program isn't doing what you want it to. Instead, ask yourself why it is doing what it is.*)
대답하기 쉬운 질문이고, 프로그램을 고치는 방법을 알아내는 첫 단계가 됨
- ❖ 버그는 당신이 생각하는 곳에 있지 않을 수 있다는 것을 명심하자. (*Keep in mind that the bug is probably not where you think it is.*)
어디를 살펴 보아야 할지 결정하는 실질적인 방법 중 하나는 버그가 어디에 있는지 묻는 것임.
(다른 모든 요소를 제거하라. 그러면 남는 한가지가 진실이다.-셜록홈즈)
- ❖ 다른 사람에게 문제를 설명해보라. (*Try to explain the problem to somebody else.*)
종종 누군가에게 문제를 설명하려고 하면 놓친 것들을 보게 된다.

Debugging 힌트

- ❖ *읽은 모든 것을 믿지 마라.(Don't believe everything you read.)*
코드가 문서의 주석(코멘트)이 제시하는 것을 수행하지 않을 수 있다.
- ❖ *디버깅을 멈추고 문서 작성을 시작하라.(Stop debugging and start writing documentation.)*
다른 관점에서 문제에 접근하는데 도움을 준다.
- ❖ *물러나서 내일 다시 시도하자.(Walk away, try again tomorrow.)*
버그를 나중에 수정한다는 의미일 수 있지만, 버그를 찾는데 소비되는 시간을 많이 줄일 수 있다.
(집착하다보면 매몰되어 더 많은 시간을 소비하게 된다. 결국 프로그래밍 문제를 해결하기 위한 일을 일찍 시작하고, 시간을 갖고 문제를 해결해나가자.)
- ❖ 버그를 발견하면 바로 수정하지 말고, 이 버그가 관측된 대부분의 오류를 설명할 수 있는지, 수정 시 다른 영향을 어떨지 잘 살펴보고, 계속 버그를 찾아서 가능하면 한번에 수정하는 것이 좋다.

파이썬

파이썬

■ 왜 파이썬(Python) ?

- 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)

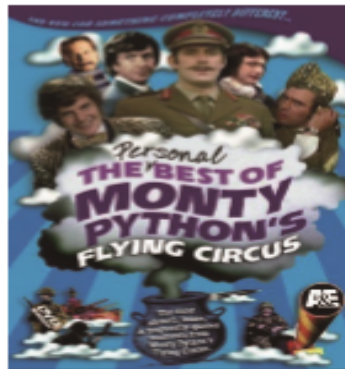


파이썬

- ❖ 1991년 귀도 반 로섬(Guido van Rossum)이 개발
 - Monty Python's 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!

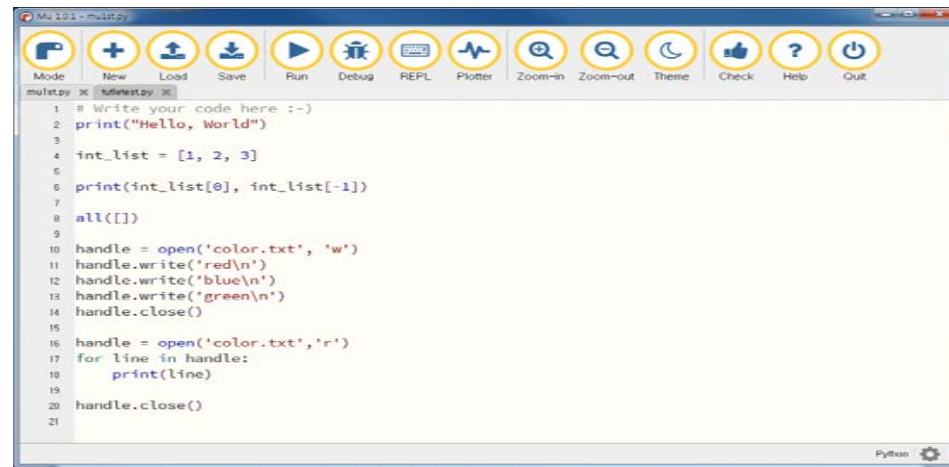
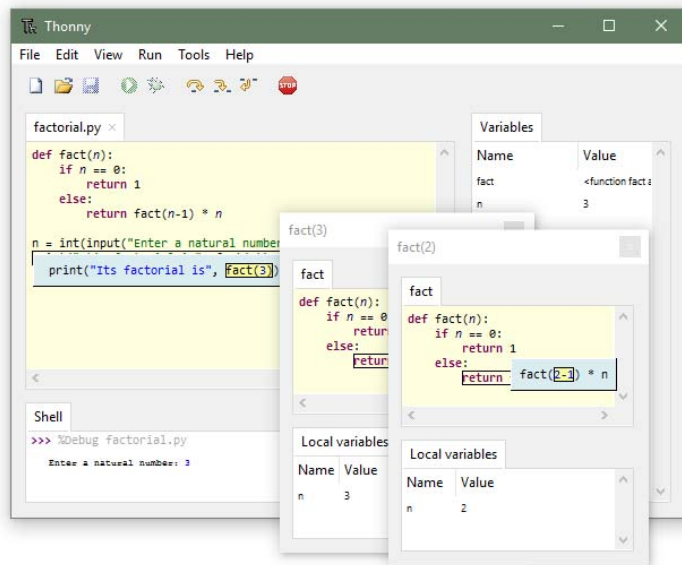


나하고는 관계가
없어!^^



파이썬

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



추천: <https://thonny.org/>

파이썬

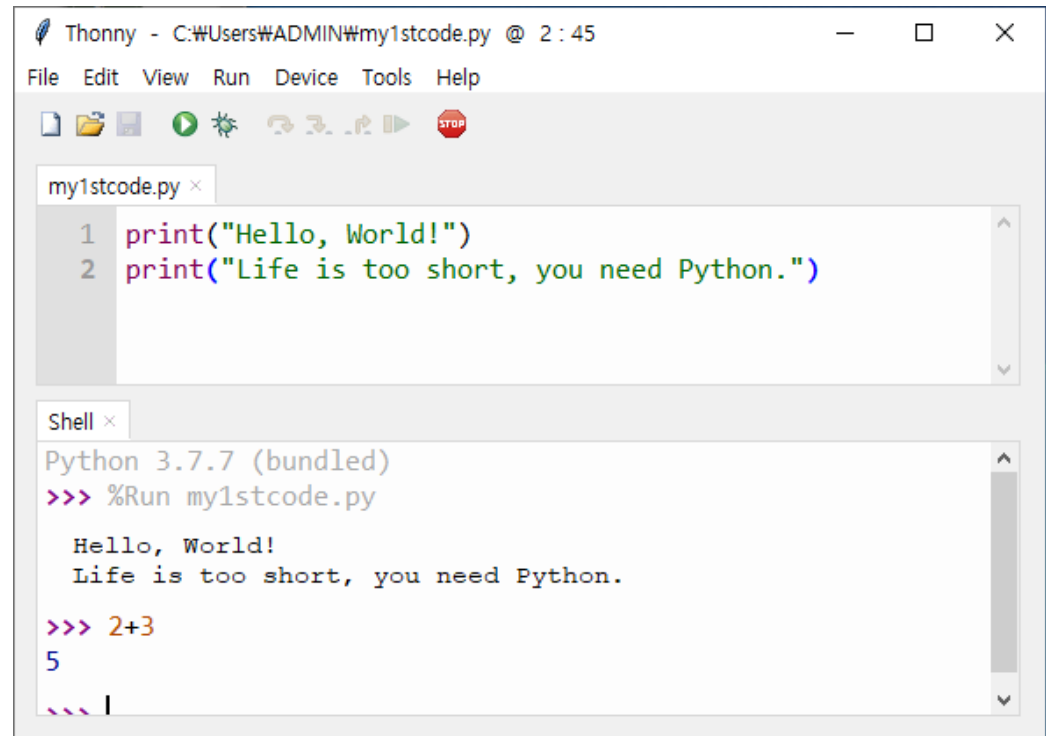
❖ 파이썬 실습

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". Below the menu is a toolbar with icons for file operations, running, and stopping. The main editor window, titled "my1stcode.py", contains two lines of Python code: `1 print("Hello, World!")` and `2 print("Life is too short, you need Python.")`. Below the editor is a "Shell" window titled "Shell x". It shows the output of the script: `Python 3.7.7 (bundled)`, `>>> %Run my1stcode.py`, followed by the printed text `Hello, World!` and `Life is too short, you need Python.`. Below this, it shows an interactive prompt `>>> 2+3` with the output `5` and a cursor at the end of a new line `>>> |`.

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

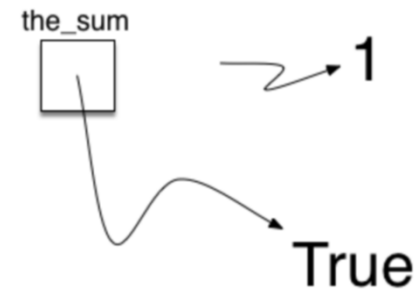
❖ 변수: 자료형을 저장하는 공간

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

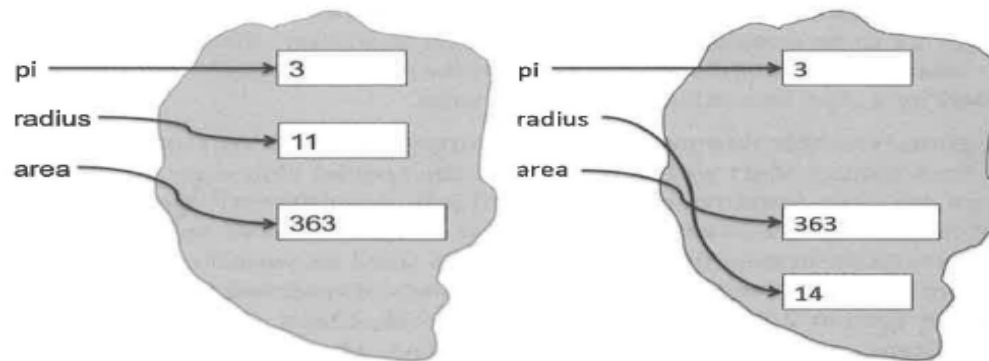


Figure 2.2 Binding of variables to objects

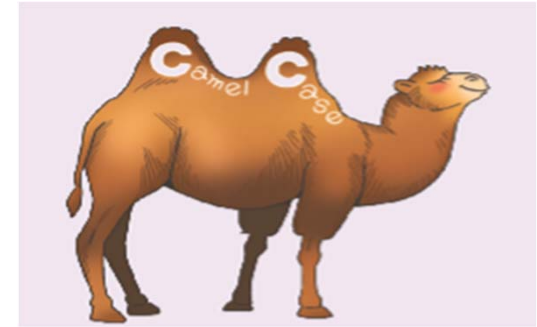
변수

❖ 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 데이터든지 저장 (다양한 자료형태 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
```

복합 자료형 (Built-in Collection Data Types)

복합 데이터 타입

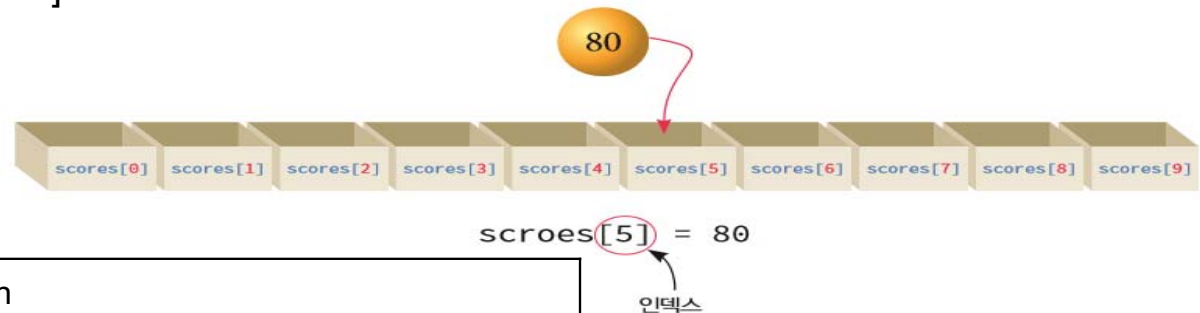
❖ Python에 사용하는 복합 자료형

- 리스트: an ordered collection of zero or more references to data objects
- 스트링: sequential collection of zero or more characters(letters, numbers and other symbols)
- 튜플: an ordered collection of zero or more references to data objects but immutable
- 집합: an unordered collection of zero or more immutable data objects
- 사전: collections of associated pairs of items where each pair consists of a key and a value

리스트

❖ list : an ordered sequence of value.

- Syntax -> [elem1, elem2, ... , elemn]
- 각 element는 어떤 type도 가능
- 각 element는 index를 통해 접근



Operations on Any Sequence in Python		
Operation Name	Operator	Explanation
Indexing	[]	Access an element of a sequence
concatenation	+	Combine sequences together
repetition	*	Concatenate a repeated number of times
membership	in	Ask whether an item is in a sequence
length	len	Ask the number of items in the sequence
slicing	[:]	Extract a part of a sequence

출처: <https://runestone.academy/runestone/books/published/pythonds3/index.html>

리스트

❖ 연산

함수나 연산자	설명	예	결과
len()	길이 계산	len([1, 2, 3])	3
+	2개의 시퀀스 연결	[1, 2] + [3, 4, 5]	[1, 2, 3, 4, 5]
*	반복	['Welcome!'] * 3	['Welcome!', 'Welcome!', 'Welcome!']
in	소속	3 in [1, 2, 3]	True
not in	소속하지 않음	5 not in [1, 2, 3]	True
[]	인덱스	myList[1]	myList의 1번째 요소
min()	시퀀스에서 가장 작은 요소	min([1, 2, 3])	1
max()	시퀀스에서 가장 큰 요소	max([1, 2, 3])	3
for 루프	반복	for x in [1, 2, 3]: print (x)	1 2 3

❖ 리스트의 인덱싱과 슬라이싱에 주의

예) [1,2,3,4][1:3][1] → 3

[1,2,3,4][1:3] → [2,3],

[2,3,4][1] → 3

[] 기호가, 리스트 표시, 슬라이스 연산, 인덱싱에 사용되므로 해석(사용)에 주의 해야함

리스트

❖ 메소드

Methods Provided by Lists in Python		
Method	Use	Explanation
append	a_list.append(item)	Adds a new item to the end of a list
insert	a_list.insert(i,item)	Inserts an item at the ith position in a list
pop	a_list.pop()	Removes and returns the last item in a list
pop	a_list.pop(i)	Removes and returns the ith item in a list
sort	a_list.sort()	Modifies a list to be sorted
reverse	a_list.reverse()	Modifies a list to be in reverse order
del	del a_list[i]	Deletes the item in the ith position
index	a_list.index(item)	Returns the index of the first occurrence of item
count	a_list.count(item)	Returns the number of occurrences of item
remove	a_list.remove(item)	Removes the first occurrence of item

```
L1 = [1,2,3]
L2 = [4,5,6]
L3 = L1 + L2
print 'L3 =', L3
L1.extend(L2)
print 'L1 =', L1
L1.append(L2)
print 'L1 =', L1
```

```
L3 = [1, 2, 3, 4, 5, 6]
L1 = [1, 2, 3, 4, 5, 6]
L1 = [1, 2, 3, 4, 5, 6, [4, 5, 6]]
```

출처: <https://runestone.academy/runestone/books/published/pythonds3/index.html>

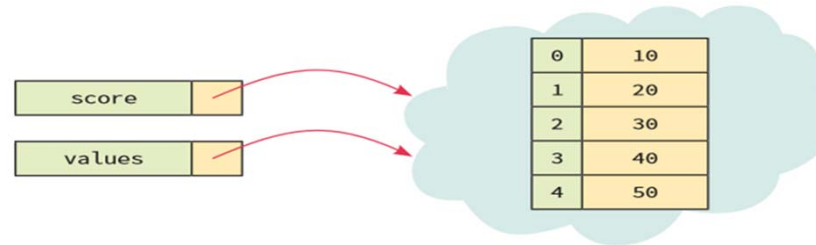
리스트 연산

연산의 예	설명
<code>mylist[2]</code>	인덱스 2에 있는 요소
<code>mylist[2] = 3</code>	인덱스 2에 있는 요소를 3으로 설정한다.
<code>del mylist[2]</code>	인덱스 2에 있는 요소를 삭제한다.
<code>len(mylist)</code>	<code>mylist</code> 의 길이를 반환한다.
<code>"value" in mylist</code>	<code>"value"</code> 가 <code>mylist</code> 에 있으면 <code>True</code>
<code>"value" not in mylist</code>	<code>"value"</code> 가 <code>mylist</code> 에 없으면 <code>True</code>
<code>mylist.sort()</code>	<code>mylist</code> 를 정렬한다.
<code>mylist.index("value")</code>	<code>"value"</code> 가 발견된 위치를 반환한다.
<code>mylist.append("value")</code>	리스트의 끝에 <code>"value"</code> 요소를 추가한다.
<code>mylist.remove("value")</code>	<code>mylist</code> 에서 <code>"value"</code> 가 나타나는 위치를 찾아서 삭제한다.

리스트 복사

❖ 얕은 복사

```
scores = [ 10, 20, 30, 40, 50 ]  
values = scores  
values[2] = 99 # scores의 값도 변환
```



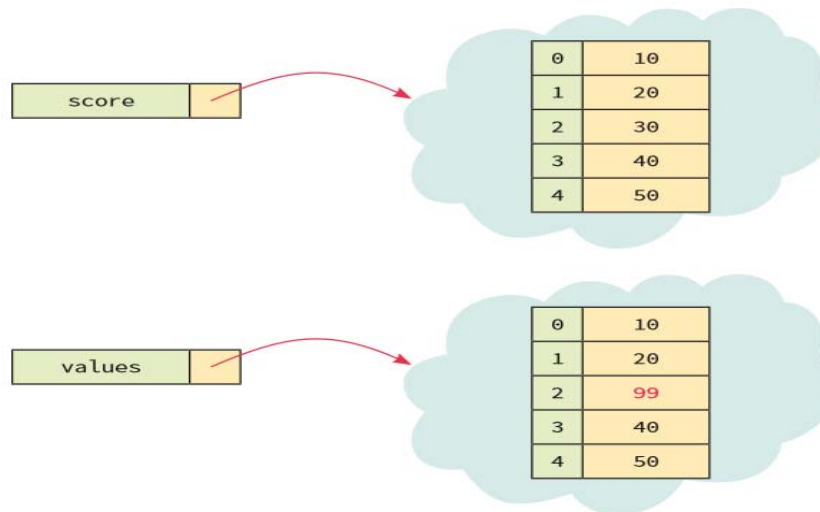
참조

❖ 깊은 복사

```
scores = [ 10, 20, 30, 40, 50 ]  
values = list(scores)  
values[2]=99
```

```
print(scores)  
print(values)
```

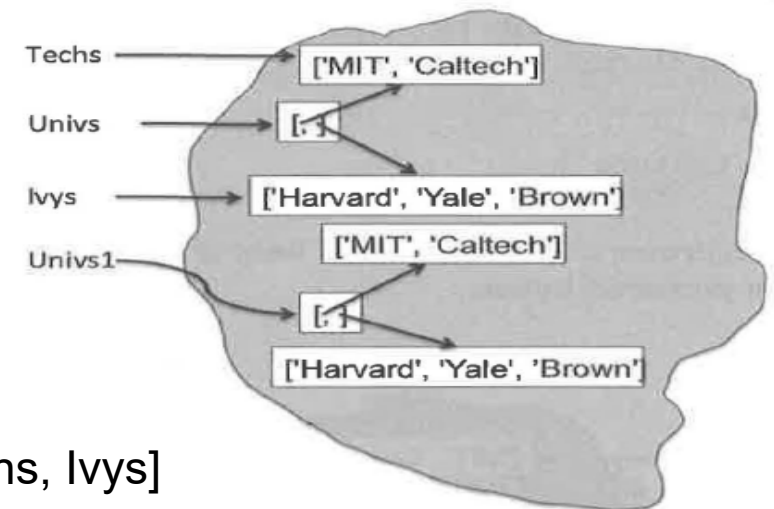
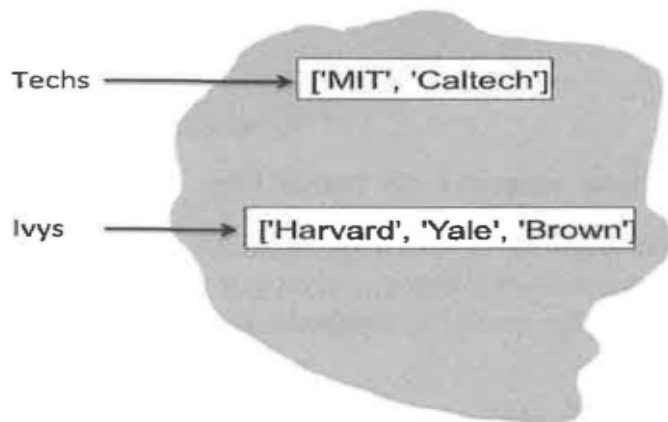
```
[10, 20, 30, 40, 50]  
[10, 20, 99, 40, 50]
```



Mutability

- ❖ 변경가능성: lists are mutable. (Tuple and strings are immutable.)
즉, 리스트는 원소를 교체할 수도 있고, 추가할 수도 있다.
 - 파이썬에서 변수는 단지 이름이다. 즉, 객체(object)에 부착된 표지(label)이다.

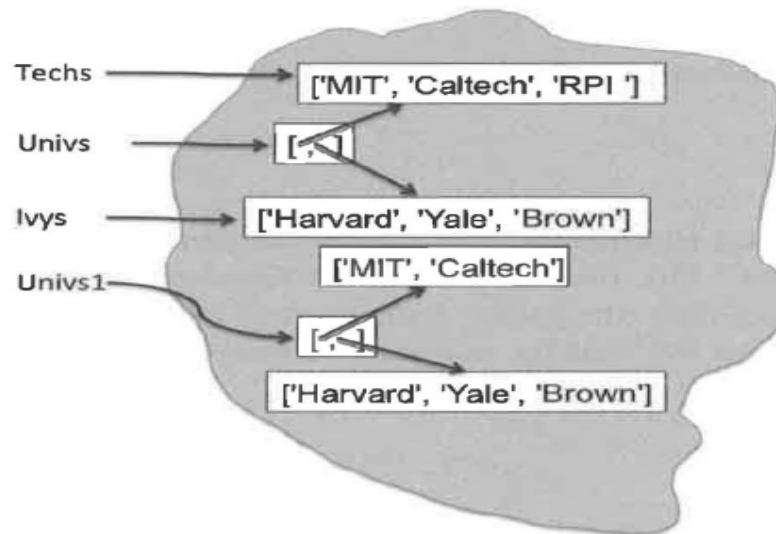
❖ Techs = ['MIT', 'Caltech']



- ❖ Univs = [Techs, Ivys]
Univs1 = [['MIT', 'Caltech'], ['Harvard', 'Yale', 'Brown']]

Mutability

- ❖ 리스트는 변경가능하기 때문에
Techs.append('RPI') -> Techs 리스트가 변형되어 RPI가 추가된다.



- ❖ Univs를 출력하게 되면 RPI가 추가된 Techs를 참조해서 리스트를 출력한다.
(aliasing 발생) → Univs엔 아무런 조작을 가하지 않았지만 Techs를 가리키고 있어서,
Techs의 변화된 내용을 참조할 수 있다.

2차원 리스트

- ❖ 행렬연산에 활용
- ❖ 2차원 테이블 표시

학생	국어	영어	수학	과학	사회
김철수	1	2	3	4	5
김영희	6	7	8	9	10
최자영	11	12	13	14	15

```
# 2차원 리스트를 생성한다.  
s = [  
    [ 1, 2, 3, 4, 5 ],  
    [ 6, 7, 8, 9, 10 ],  
    [11, 12, 13, 14, 15 ]  
]  
print(s)
```

```
[[1, 2, 3, 4, 5], [6, 7, 8, 9, 10], [11, 12, 13, 14, 15]]
```

스트링

- ❖ Type str: string을 의미
single or double quote 로 감싸인 문자열
예) 'abc' or "abc"
연속적인 큰 따옴표 3개로 구성된
문자열은 다중라인을 표시한다.

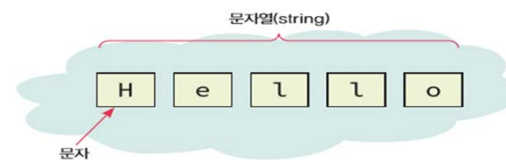
““““

동해물과 백두산이 마르고 닳도록
하느님이 보우하사 우리나라 만세

””””

- ❖ '123' => 수 123이 아니고 문자열
- ❖ +, *을 string 연산에 사용할 수 있음
>>> 'a'
>>> 3*4
>>> 3*'a'
>>> 'a'+'a'

문자열은 문자들의 순서있는 집합



문자열은 문자들의
순서있는 집합입니다.



문자 하나 하나가 1바이트씩 할당되어 있음.

P	y	t	h	o	n
0	1	2	3	4	5

인덱스는 문자에 매겨진
번호입니다. 0부터 시작해요!



스트링

- ❖ Type checking: type 간의 연산이 가능한지 조사

str*str : 오류

'4' < 3 => F or T (파이썬은 type checking 강하지 않아서

이런 것은 허용함, C와 유사 => static semantic)

- ❖ Length of string len(): len('abc') => 3

- ❖ Indexing: 스트링내의 개별 문자를
참조할 경우? 0부터 index가 시작됨

>>> 'abc'[0]	a	b	c
>>> 'abc'[3]	0	1	2
>>> 'abc'[-1]	-3	-2	-1

Type conversions (or type cast)

타입을 바꾸는 연산

int('3')*4 → 12

int(3.9) → 3

- ❖ Slicing: 문자열의 일부만 추출

s[start:end] => 문자열 s의 start위치부터 end-1까지의 부분문자열

>>> 'abc'[1:3] => 'bc'

>>> 'abc'[:2] => 'ab' # 시작값은 0이 기본값임

>>> 'abc'[:] => 'abc'

스트링

❖ 메소드

s.count(s1) counts how many times the string s1 occurs in s.

s.find(s1) returns the index of the first occurrence of the substring s1 in s, and -1 if s1 is not in s.

s.rfind(s1) same as find, but starts from the end of s (the “r” in rfind stands for reverse).

s.index(s1) same as find, but raises an exception (see Chapter 7) if s1 is not in s.

s.rindex(s1) same as index, but starts from the end of s.

s.lower() converts all uppercase letters in s to lowercase.

s.replace(old, new) replaces all occurrences of the string old in s with the string new.

s.rstrip() removes trailing white space from s.

s.split(d) Splits s using d as a delimiter. Returns a list of substrings of s. For example, the value of 'David Gutttag plays basketball'.split(' ') is ['David', 'Gutttag', 'plays', 'basketball']. If d is omitted, the substrings are separated by arbitrary strings of whitespace characters (space, tab, newline, return, and formfeed).

Method	Use	Explanation
center	a_string.center(w)	Returns a string centered in a field of size w
ljust	a_string.ljust(w)	Returns a string left-justified in a field of size w
rjust	a_string.rjust(w)	Returns a string right-justified in a field of size w

튜플

❖ 튜플(Tuple) : ordered sequences of elements

- Syntax -> (elem1, elem2, ... , elemn)
- 각 element는 어떤 type도 가능

```
t1 = (1, 'two', 3)
t2 = (t1, 3.25)
print t2
print (t1 + t2)
print (t1 + t2)[3]
print (t1 + t2)[2:5]
((1, 'two', 3), 3.25)
(1, 'two', 3, (1, 'two', 3), 3.25)
(1, 'two', 3)
(3, (1, 'two', 3), 3.25)
```

다중할당

- ❖ sequence 자료 (tuple, string)에서 각 원소를 추출해서 할당

x, y = (3, 4) → x = 3, y = 4

a, b, c = 'xyz' → a = x, b = y, c = z

- ❖ 사용 : minDivisor, maxDivisor = findExtremeDivisors(100,20)

→ minDivisor = 20, maxDivisor = 100

```
def findExtremeDivisors(n1, n2):  
    """Assumes that n1 and n2 are positive ints  
    Returns a tuple containing the smallest common  
    divisor > 1 and the largest common divisor of n1  
    and n2"""  
    divisors = () #the empty tuple  
    minVal, maxVal = None, None  
    for i in range(2, min(n1, n2) + 1):  
        if n1%i == 0 and n2%i == 0:  
            if minVal == None or i < minVal:  
                minVal = i  
            if maxVal == None or i > maxVal:  
                maxVal = i  
    return (minVal, maxVal)
```



Strings, Tuples, and Lists

❖ Sequence type: str, tuple, list => 공통 연산 및 비교

`seq[i]` returns the i^{th} element in the sequence.
`len(seq)` returns the length of the sequence.
`seq1 + seq2` returns the concatenation of the two sequences.
`n * seq` returns a sequence that repeats `seq` n times.
`seq[start:end]` returns a slice of the sequence.
`e in seq` is True if `e` is contained in the sequence and False otherwise.
`e not in seq` is True if `e` is not in the sequence and False otherwise.
`for e in seq` iterates over the elements of the sequence.

Type	Type of elements	Examples of literals	Mutable
str	characters	<code>' '</code> , <code>'a'</code> , <code>'abc'</code>	No
tuple	any type	<code>()</code> , <code>(3,)</code> , <code>('abc', 4)</code>	No
list	any type	<code>[]</code> , <code>[3]</code> , <code>['abc', 4]</code>	Yes

※ 계산도중 내용을 추가할 수 있는 list가 많이 사용된다.

집합

❖ 집합(set) : an unordered collection of zero or more immutable data objects

- Syntax -> { elem1, elem2, ... , elemn } // do not allow duplicates
- 각 element는 어떤 type도 가능

Operation	Operator	Explanation
membership	in	Set membership
length	len	Returns the cardinality of the set
	a_set other_set	Returns a new set with all elements from both sets
&	a_set & other_set	Returns a new set with only those elements common to both sets
-	a_set - other_set	Returns a new set with all items from the first set not in second
<=	a_set <= other_set	Asks whether all elements of the first set are in the second

집합

Method Name	Use	Explanation
union	<code>a_set.union(other_set)</code>	Returns a new set with all elements from both sets
intersection	<code>a_set.intersection(other_set)</code>	Returns a new set with only those elements common to both sets
difference	<code>a_set.difference(other_set)</code>	Returns a new set with all items from first set not in second
issubset	<code>a_set.issubset(othe_rset)</code>	Asks whether all elements of one set are in the other
add	<code>a_set.add(item)</code>	Adds item to the set
remove	<code>a_set.remove(item)</code>	Removes item from the set
pop	<code>a_set.pop()</code>	Removes an arbitrary element from the set
clear	<code>a_set.clear()</code>	Removes all elements from the set

사전

- ❖ 사전(dictionaries) → a set of key/value pair

정수로 인덱싱하지 않고, key로 인덱싱해서 원하는 value를 찾는다.

```
monthNumbers = {'Jan':1, 'Feb':2, 'Mar':3, 'Apr':4, 'May':5,  
                1:'Jan', 2:'Feb', 3:'Mar', 4:'Apr', 5:'May'}  
print 'The third month is ' + monthNumbers[3]  
dist = monthNumbers['Apr'] - monthNumbers['Jan']  
print 'Apr and Jan are', dist, 'months apart'
```

```
The third month is Mar  
Apr and Jan are 3 months apart
```



- ❖ monthNumbers[3] 에서 3은 key값 3을 의미한다.
(정수로 인덱싱해서 사전의 네번째 원소를 의미하는 것이 아니다.)
- ❖ print monthNumbers.keys()

→ [1, 2, 'Mar', 'Feb', 5, 'Apr', 'Jan', 'May', 3, 4]

사전

❖ 사전에 사용되는 연산과 메소드

`len(d)` returns the number of items in `d`.
`d.keys()` returns a list containing the keys in `d`.
`d.values()` returns a list containing the values in `d`.
`k in d` returns `True` if key `k` is in `d`.
`d[k]` returns the item in `d` with key `k`.
`d.get(k, v)` returns `d[k]` if `k` is in `d`, and `v` otherwise.
`d[k] = v` associates the value `v` with the key `k` in `d`. If there is already a value associated with `k`, that value is replaced.
`del d[k]` removes the key `k` from `d`.
`for k in d` iterates over the keys in `d`.

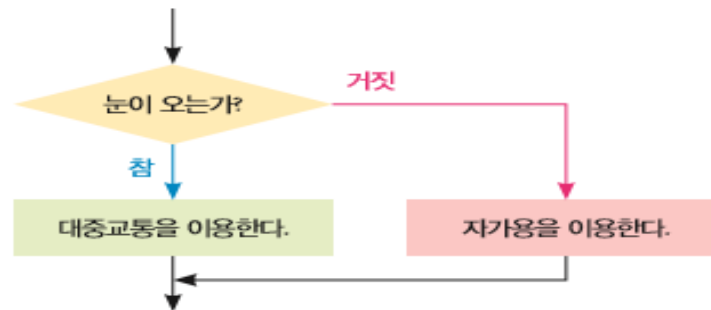
Method	Use	Explanation
keys	<code>a_dict.keys()</code>	Returns the keys of the dictionary in a <code>dict_keys</code> object
values	<code>a_dict.values()</code>	Returns the values of the dictionary in a <code>dict_values</code> object
items	<code>a_dict.items()</code>	Returns the key-value pairs in a <code>dict_items</code> object
get	<code>a_dict.get(k)</code>	Returns the value associated with <code>k</code> , <code>None</code> otherwise
get	<code>a_dict.get(k, alt)</code>	Returns the value associated with <code>k</code> , <code>alt</code> otherwise

Python 기초

선택과 반복

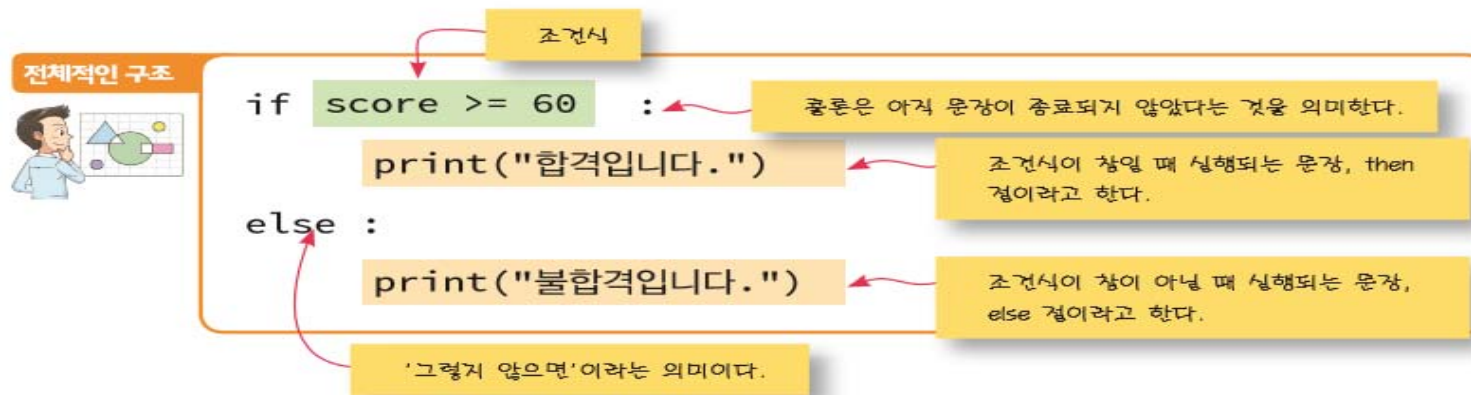
if ~ else 문

- ❖ 조건의 참/거짓에 따라 할 일을 선택



- ❖ Syntax

- 참 영역이나 거짓영역의 하위 블록을 표시하기 위해 꼭 들여쓰기를 해야함



조건식

❖ if 문안의 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?

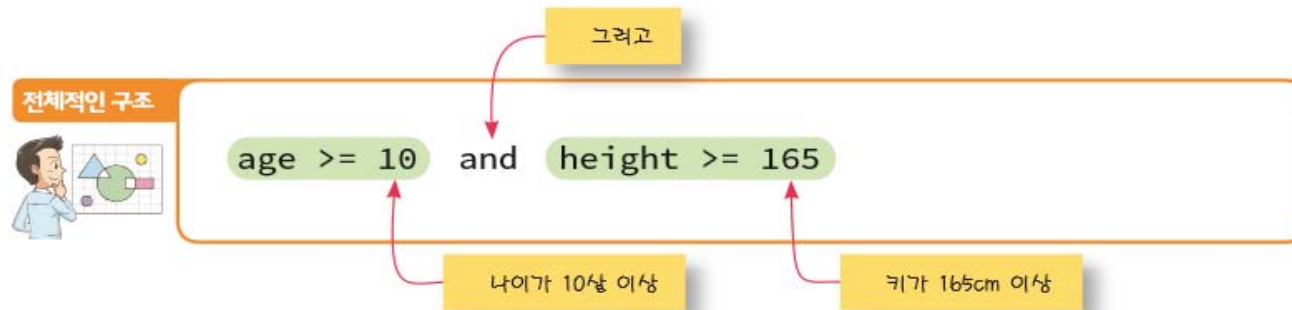
C: block을 { c_statements; }

Python: indentation level로 블록표현

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

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
```

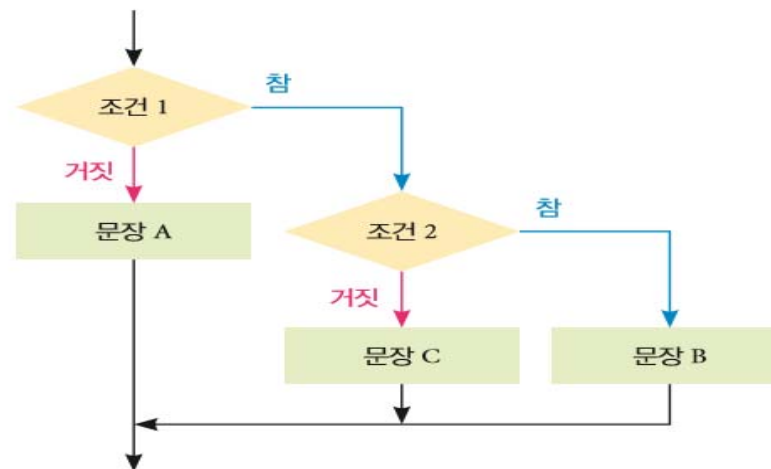
```
else:
```

```
    if 조건2 :
```

```
        문장_B
```

```
    else:
```

```
        문장_C
```



if ~ else의 내포

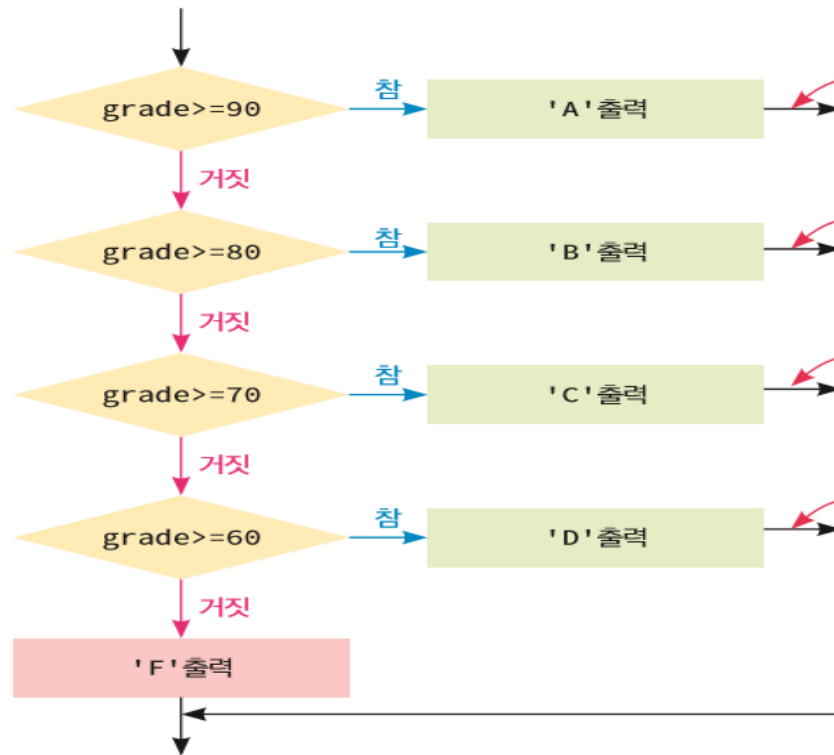
- ❖ 사과를 신선하다면 사기로 했다. 그런데 신선한 사과의 가격이 1000원 미만이면 10개를 사고, 1000원 이상이면 5개를 사고자 한다.

```
appleQuality = input("사과의 상태를 입력하시오: ")
applePrice = int(input("사과 1개의 가격을 입력하시오: "))

if appleQuality == "신선":
    if applePrice < 1000:
        print("10개를 산다")
    else:
        print("5개를 산다")
else:
    print("사과를 사지 않는다.")
```

if ~ elif ~ ~ else 구조

❖ 조건구간의 세분화 (elif : else if 의 축약형)

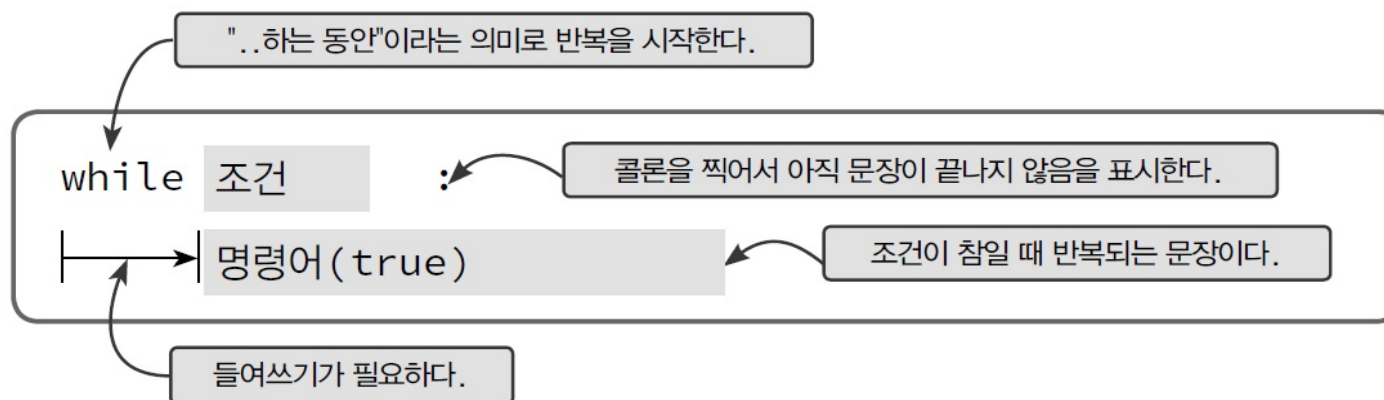
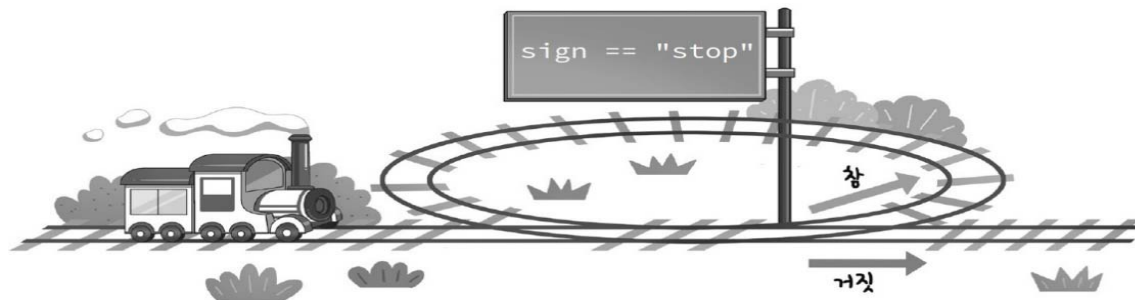


```
grade = int(input("성적을 입력하시오: "))
```

```
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

```
>>> 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
```

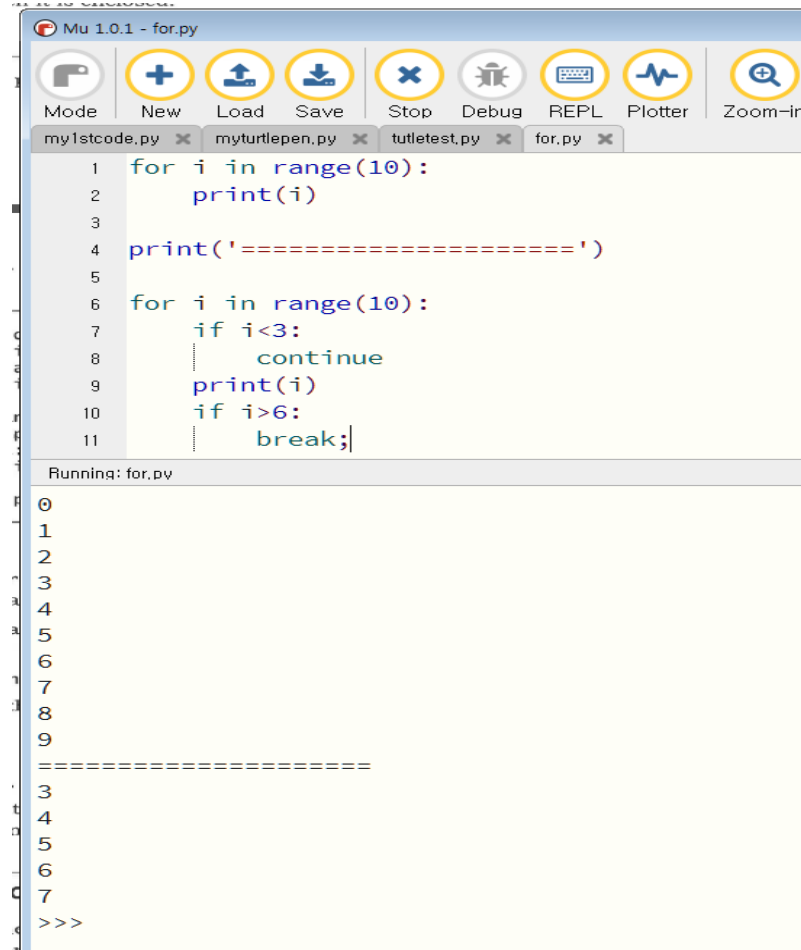
```
8
```

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



The screenshot shows the Mu Python IDE interface. The top toolbar includes icons for Mode, New, Load, Save, Stop, Debug, REPL, Plotter, and Zoom-in. The file explorer shows several files: my1stcode.py, myturtlepen.py, tutletest.py, and for.py. The main editor displays a Python script with the following code:

```
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;
```

The output window shows the execution results:

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

파이썬 코딩

Covid-19 관련 예제

- ❖ How I use Python to map the global spread of COVID-19
(<https://opensource.com/article/20/4/python-map-covid-19>)
 - Python과 Pandas를 활용하여 Covid-19의 확산을 시각화
 - Pandas : 파이썬 데이터 분석 라이브러리