

| Python review

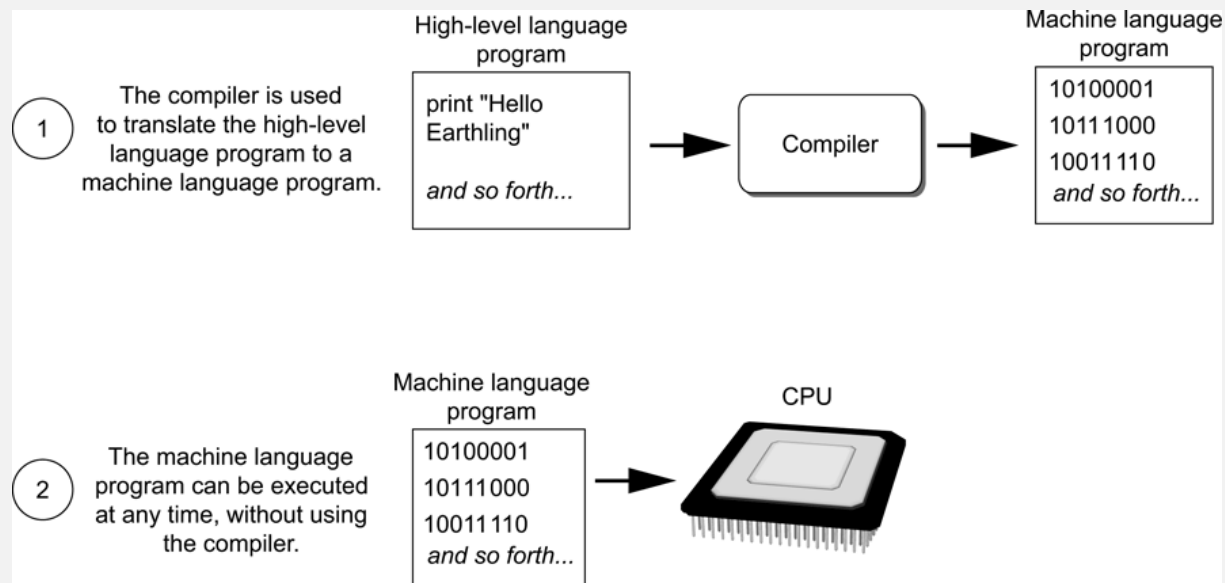


명지대학교
MYONGJI UNIVERSITY

컴파일러와 인터프리터

컴파일러 :

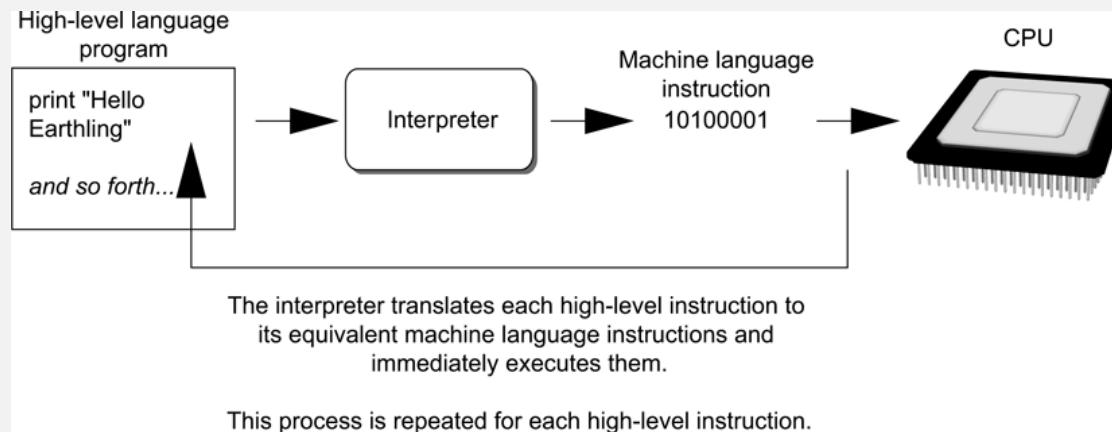
- High-level 언어를 Machine Language 로 변경해야 한다
- Ex) C, C++,



컴파일러와 인터프리터

인터프리터

- Ex) Python
- 한번에 하나의 명령어만 가능하다
- Machine language 프로그램과 구분되지 않는다



Python Language Basics, IPython, and Jupyter Notebooks

\$python

```
Python 3.7.3 (default, Apr 24 2020, 18:51:23)
[Clang 11.0.3 (clang-1103.0.32.62)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 1
>>> print(a)
1
>>>
```

\$ipython

```
Python 3.7.3 (default, Apr 24 2020, 18:51:23)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.15.0 -- An enhanced Interactive Python. Type '?' for help.
```

```
In [1]: print("Hello world")
Hello world
```

```
In [2]:
```

```
$ python hello_world.py
Hello world
```

If else 문 예제

반지름을 입력 받아 그 숫자가 0보다 크면 부피를 구하고, 0 보다 작은 수가 입력되었을 때는 음수라고 작성한다.

```
radius = eval(input("Enter radius"))
if radius >= 0:
    area = radius * radius * 3.1415
    print("The area for the circle of radius", radius, "is", area)
else:
    print("Negative input")
```

```
Enter radius56
The area for the circle of radius 56 is
9851.744
```

```
Enter radius-10
Negative input
```

지역 변수(local variable)

- **Local variable(지역변수)**: 함수 안에서만 사용하는 변수
 - 함수에서 생성된 변수
 - 생성된 함수에서만 접근 가능하고 다른 함수에서 접근 하려 하면 에러
- **Scope(범위)**: 변수 사용이 가능한 프로그램의 부분
 - 지역변수의 Scope 는 생성된 함수 안
- **다른 함수는 같은 지역변수 이름을 이어도 서로 영향을 미치지 않음**
 - 각 함수는 다른 함수의 지역변수 값에 접근 불가

지역 변수 사용 잘못된 예

```
# Definition of the main function.
def main():
    get_name()
    print('Hello', name)      # This causes an error!

# Definition of the get_name function.
def get_name():
    name = input('Enter your name: ')

# Call the main function.
main()
|
```

Enter your name: Hong Kil Dong
Traceback (most recent call last):
This causes an error!
NameError: name 'name' is not defined

함수에서 Argument 넘겨주기

```
# This program demonstrates an argument being
# passed to a function.

def main():
    value = 5
    show_double(value)

# The show_double function accepts an argument
# and displays double its value.
def show_double(number):
    result = number * 2
    print(result)

# Call the main function.
main()
```

10

Lists 소개

- **List: 다양한 데이터가 모아져 있는 객체**
 - Element(원소): 리스트의 하나하나 아이템
 - Format: `list = [item1, item2, etc.]`
 - 다른 타입의 원소 가능
- **print 함수는 리스트 전체를 보여줌**
- **list() 함수는 리스트 객체로 변환**

인덱싱

- **Index: 리스트의 원소의 특별한 위치 지정**

- 리스트의 각 원소값을 지정하여 접근 가능
- 리스트의 첫번째 Index 값은 0,
- 리스트의 두번째 index 값은 1
- N번째의 index 값은 n-1
- 마이너스 값은 맨 마지막부터 사용
 - index -1 값은 리스트 원소의 맨 마지막 의미
 - Index -2 값은 리스트 원소의 맨 마지막에서 두번째 의미

```
>>> list = [ 1 , 9 , 10, 3 , 8]
>>> list[0]
1
>>> list[-1]
8
>>> list[-3]
10
```

Lists 변경가능

- 원소값 변경 가능 시퀀스: 원소값의 인덱스 값으로 접근하여 값 변경 가능
 - Lists의 원소 값을 변경 가능
 - 단 유효하지 않은 원소 인덱스 접근시 `IndexError` 발생

```
>>> list_val
[10, 20, 30, 40]
>>> list_val[0] = 100
>>> list_val
[100, 20, 30, 40]
>>> list_val[5] = 1000
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list assignment index out of range
```

Lists 합치기

- Concatenate(합치기): join two things together
- + 연산자는 두개의 리스트 더하기 가능
 - List이외 다른 데이터(ex : int) 타입과 + 로 더하기는 불가능

```
>>> list_val = [100, 20, 30, 40]
>>> list_val2 = [ 100,200,300]
>>> list_val + list_val2
[100, 20, 30, 40, 100, 200, 300]
>>> list_val2 + list_val
[100, 200, 300, 100, 20, 30, 40]
```

```
>>> list_val2 =[ 100,200,300]
>>> list_val2 + 4
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate list (not "int") to list
```

list

숫자

리스트 자르기

- **Slice**: 시퀀스 리스트의 특정 아이템 값만 뽑아내기

- List slicing format: `list[start : end]`
- list 변수에서 `start` 부터 시작하지만 `end` 는 포함하지 않음
 - 만약 시작 지점이 지정되지 않으면 기본값으로 0
 - 만약 끝 지점이 지정되지 않으면 기본값으로 `len(list)`
- 특정 범위를 지정하여 리스트 값 가지고오기

```
>>> list_val = [ 4, 5, 10, 99, -1, 33, 111]
>>> list_val[:3]
[4, 5, 10]
>>> list_val[3:5]
[99, -1]
```

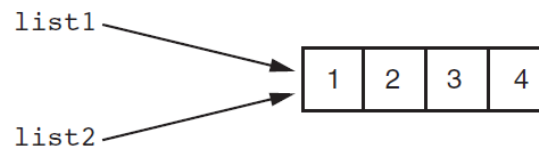
리스트 복사

- 리스트의 원소값을 복사

- 다음과 같은 두가지 방법으로 복사가능:

- 빈 리스트 생성 후 for 문을 이용하여 원소 하나씩 넣는 방법
- 빈 리스트 생성 후 이전 리스트를 연결

Figure 7-4 list1 and list2 reference the same list



```
>>> list1=[1,2,3,4,5]
>>> list2=[]
>>> for i in range(len(list1)):
>>>     list2.insert(i, list1[i])
```

```
>>> list2
[1, 2, 3, 4, 5]
>>> list3=[]
>>> list3 = list1
>>> list3
[1, 2, 3, 4, 5]
```

이차원 리스트

Figure 7-7 Subscripts for each element of the `scores` list

	Column 0	Column 1	Column 2
Row 0	<code>scores[0][0]</code>	<code>scores[0][1]</code>	<code>scores[0][2]</code>
Row 1	<code>scores[1][0]</code>	<code>scores[1][1]</code>	<code>scores[1][2]</code>
Row 2	<code>scores[2][0]</code>	<code>scores[2][1]</code>	<code>scores[2][2]</code>

리스트

List

```
In [24]: a_list = [2, 3, 7, None]
          tup = ('foo', 'bar', 'baz')
          b_list = list(tup)
          b_list
          b_list[1] = 'peekaboo'
          b_list
```

```
Out[24]: ['foo', 'peekaboo', 'baz']
```

```
In [25]: gen = range(10)
          gen
          list(gen)
```

```
Out[25]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```


리스트

Adding and removing elements

```
In [26]: b_list.append('dwarf')  
b_list
```

```
Out[26]: ['foo', 'peekaboo', 'baz', 'dwarf']
```

```
In [27]: b_list.insert(1, 'red')  
b_list
```

```
Out[27]: ['foo', 'red', 'peekaboo', 'baz', 'dwarf']
```

```
In [28]: b_list.pop(2)  
b_list
```

```
Out[28]: ['foo', 'red', 'baz', 'dwarf']
```

```
In [29]: b_list.append('foo')  
b_list  
b_list.remove('foo')  
b_list
```

```
Out[29]: ['red', 'baz', 'dwarf', 'foo']
```

```
In [30]: 'dwarf' in b_list
```

```
Out[30]: True
```

```
In [31]: 'dwarf' not in b_list
```

```
Out[31]: False
```

리스트

리스트 이어 붙이기

Concatenating and combining lists

```
In [32]: [4, None, 'foo'] + [7, 8, (2, 3)]
```

```
Out[32]: [4, None, 'foo', 7, 8, (2, 3)]
```

```
In [33]: x = [4, None, 'foo']  
x.extend([7, 8, (2, 3)])  
x
```

```
Out[33]: [4, None, 'foo', 7, 8, (2, 3)]
```

```
everything = []  
for chunk in list_of_lists:  
    everything.extend(chunk)
```

```
everything = []  
for chunk in list_of_lists:  
    everything = everything + chunk
```

리스트

정렬

Sorting

```
In [34]: a = [7, 2, 5, 1, 3]
         a.sort()
         a
```

```
Out[34]: [1, 2, 3, 5, 7]
```

```
In [35]: b = ['saw', 'small', 'He', 'foxes', 'six']
         b.sort(key=len)
         b
```

```
Out[35]: ['He', 'saw', 'six', 'small', 'foxes']
```

Binary search and maintaining a sorted list

```
In [36]: import bisect
         c = [1, 2, 2, 2, 3, 4, 7]
         bisect.bisect(c, 2)
         bisect.bisect(c, 5)
         bisect.insort(c, 6)
         c
```

```
Out[36]: [1, 2, 2, 2, 3, 4, 6, 7]
```

리스트

Slicing

```
In [37]: seq = [7, 2, 3, 7, 5, 6, 0, 1]
         seq[1:5]
```

```
Out[37]: [2, 3, 7, 5]
```

```
In [38]: seq[3:4] = [6, 3]
         seq
```

```
Out[38]: [7, 2, 3, 6, 3, 5, 6, 0, 1]
```

```
In [39]: seq[:5]
         seq[3:]
```

```
Out[39]: [6, 3, 5, 6, 0, 1]
```

```
In [40]: seq[-4:]
         seq[-6:-2]
```

```
Out[40]: [6, 3, 5, 6]
```

```
In [41]: seq[::2]
```

```
Out[41]: [7, 3, 3, 6, 1]
```

```
In [42]: seq[::-1]
```

```
Out[42]: [1, 0, 6, 5, 3, 6, 3, 2, 7]
```

내장 순차 자료형 함수

enumerate

```
In [43]: some_list = ['foo', 'bar', 'baz']  
mapping = {}  
for i, v in enumerate(some_list):  
    mapping[v] = i  
mapping
```

```
Out[43]: {'foo': 0, 'bar': 1, 'baz': 2}
```

내장 순차 자료형 함수

sorted

```
In [44]: sorted([7, 1, 2, 6, 0, 3, 2])  
sorted('horse race')
```

```
Out[44]: [' ', 'a', 'c', 'e', 'e', 'h', 'o', 'r', 'r', 's']
```

reversed

```
In [49]: list(reversed(range(10)))
```

```
Out[49]: [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

내장 순차 자료형 함수

zip

```
In [45]: seq1 = ['foo', 'bar', 'baz']  
seq2 = ['one', 'two', 'three']  
zipped = zip(seq1, seq2)  
list(zipped)
```

```
Out[45]: [('foo', 'one'), ('bar', 'two'), ('baz', 'three')]
```

```
In [46]: seq3 = [False, True]  
list(zip(seq1, seq2, seq3))
```

```
Out[46]: [('foo', 'one', False), ('bar', 'two', True)]
```

```
In [47]: for i, (a, b) in enumerate(zip(seq1, seq2)):  
        print('{0}: {1}, {2}'.format(i, a, b))
```

```
0: foo, one  
1: bar, two  
2: baz, three
```

```
In [48]: pitchers = [('Nolan', 'Ryan'), ('Roger', 'Clemens'),  
                    ('Schilling', 'Curt')]  
first_names, last_names = zip(*pitchers)  
first_names  
last_names
```

```
Out[48]: ('Ryan', 'Clemens', 'Curt')
```

내장 순차 자료형 함수

reversed

```
In [49]: list(reversed(range(10)))
```

```
Out[49]: [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```


| Python review 2



명지대학교
MYONGJI UNIVERSITY

튜플(Tuples)

- **Tuple: 변경 불가능한 시퀀스**

- 리스트와 유사
- 한번 생성하면 변경 불가
- Format: `tuple_name = (item1, item2)`
- 리스트처럼 지원 가능한 연산자
 - 각 원소값들의 indexing
 - `len`, `min`, `max` 와 같은 내장 함수들
 - 자르기
 - `in`, `+`, `and` * 연산자 가능

튜플(Tuples)

- **Tuples은 다음과 같은 메소드를 제공하지 않음**
 - append
 - remove
 - insert
 - reverse
 - sort

튜플(Tuples)

```
In [7]: tup = 4, 5, 6  
tup
```

```
Out[7]: (4, 5, 6)
```

```
In [8]: nested_tup = (4, 5, 6), (7, 8)  
nested_tup
```

```
Out[8]: ((4, 5, 6), (7, 8))
```

```
In [9]: tuple([4, 0, 2])  
tup = tuple('string')  
tup
```

```
Out[9]: ('s', 't', 'r', 'i', 'n', 'g')
```

```
In [10]: tup[0]
```

```
Out[10]: 's'
```

```
In [11]: tup = tuple(['foo', [1, 2], True])  
tup[2] = False
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-11-11b694945ab9> in <module>  
      1 tup = tuple(['foo', [1, 2], True])  
----> 2 tup[2] = False  
  
TypeError: 'tuple' object does not support item assignment
```

튜플(Tuples)

```
In [14]: tup[1].append(3)  
tup
```

```
Out[14]: ('foo', [1, 2, 3], True)
```

```
In [15]: (4, None, 'foo') + (6, 0) + ('bar',)
```

```
Out[15]: (4, None, 'foo', 6, 0, 'bar')
```

```
In [16]: ('foo', 'bar') * 4
```

```
Out[16]: ('foo', 'bar', 'foo', 'bar', 'foo', 'bar', 'foo', 'bar')
```

튜플(Tuples)

Unpacking tuples

```
In [17]: tup = (4, 5, 6)
         a, b, c = tup
         b
```

Out[17]: 5

```
In [18]: tup = 4, 5, (6, 7)
         a, b, (c, d) = tup
         d
```

Out[18]: 7

```
tmp = a
a = b
b = tmp
```

```
In [19]: a, b = 1, 2
         a
         b
         b, a = a, b
         a
         b
```

Out[19]: 1

```
In [20]: seq = [(1, 2, 3), (4, 5, 6), (7, 8, 9)]
         for a, b, c in seq:
             print('a={0}, b={1}, c={2}'.format(a, b, c))
```

```
a=1, b=2, c=3
a=4, b=5, c=6
a=7, b=8, c=9
```

```
In [21]: values = 1, 2, 3, 4, 5
         a, b, *rest = values
         a, b
         rest
```

Out[21]: [3, 4, 5]

```
In [22]: a, b, *_ = values
```

튜플(Tuples)

Tuple methods

```
In [23]: a = (1, 2, 2, 2, 3, 4, 2)  
         a.count(2)
```

```
Out[23]: 4
```

딕셔너리(Dictionary)

- **Dictionary:** 데이터를 저장 할 수 있는 객체
- 사람은 누구든지 "이름" = "홍길동", "생일" = "몇 월 몇 일" 등으로 구분할 수 있다. 파이썬은 영리하게도 이러한 대응 관계를 나타낼 수 있는 자료형을 가지고 있다.
 - 모든 원소는 키 와 값을 가지고 있다.
 - 키와 값의 매핑이라고 이야기 하기도 함
 - 반드시 변경 불가능한 객체 여야 함
 - 어떠한 값을 알기 위해서 그와 관련된 키 값을 알아야 한다.
 - 딕셔너리 포맷
 - `dictionary =`
`{key1:val1, key2:val2}`

딕셔너리(Dictionary)로 부터 값 가지고 오기

- 딕셔너리의 원소는 정렬되어 있지 않음
- `dictionary[key]` 와 같은 포맷으로 딕셔너리내의key 값을 가지고 있는 값들을 나타낼 수 있음
 - 만약 딕셔너리안에서 key 값이 없다면, `KeyError`에러 발생
- 딕셔너리 안에서 키 값이 있는지 없는 지 확인을 위해서 `in` 과 `not in` 연산자를 이용

```
name_age = {}  
name_age = {"alice":5, "bob":7, "Grace":9}  
print(name_age["alice"])
```

5

이미 있는 딕셔너리에 추가

- 딕셔너리 자체는 변경가능 한 객체이다
- 키와 값의 쌍으로 추가 하기 위해서:

dictionary[key] = value

- 만약 키 값이 이미 존재하고 있다면 , 그 값을 바꾼다

```
name_age = {}  
name_age = {"alice":5, "bob":7, "Grace":9}  
print(name_age["alice"])  
name_age["alice"] = 10  
print(name_age)  
name_age["kim"] = 15  
print(name_age)
```

```
5  
{'alice': 10, 'bob': 7, 'Grace': 9}  
{'alice': 10, 'bob': 7, 'Grace': 9, 'kim':  
15}
```

```
age={}  
age["kim"] = 10  
print(age)  
age["Lee"] = 20  
print(age)  
age["park"] = 30  
print(age)
```

```
{'kim': 10}  
{'kim': 10, 'Lee': 20}  
{'kim': 10, 'Lee': 20, 'park':  
30}
```

딕셔너리(Dictionary)

```
In [50]: empty_dict = {}  
d1 = {'a' : 'some value', 'b' : [1, 2, 3, 4]}  
d1
```

```
Out[50]: {'a': 'some value', 'b': [1, 2, 3, 4]}
```

```
In [51]: d1[7] = 'an integer'  
d1  
d1['b']
```

```
Out[51]: [1, 2, 3, 4]
```

```
In [52]: 'b' in d1
```

```
Out[52]: True
```

```
In [53]: d1[5] = 'some value'  
d1  
d1['dummy'] = 'another value'  
d1  
del d1[5]  
d1  
ret = d1.pop('dummy')  
ret  
d1
```

```
Out[53]: {'a': 'some value', 'b': [1, 2, 3, 4], 7: 'an integer'}
```

```
In [54]: list(d1.keys())  
list(d1.values())
```

```
Out[54]: ['some value', [1, 2, 3, 4], 'an integer']
```

```
In [55]: d1.update({'b' : 'foo', 'c' : 12})  
d1
```

```
Out[55]: {'a': 'some value', 'b': 'foo', 7: 'an integer', 'c': 12}
```

딕셔너리(Dictionary)

Creating dicts from sequences

mapping = {} for key, value in zip(key_list, value_list): mapping[key] = value

```
In [56]: mapping = dict(zip(range(5), reversed(range(5))))  
mapping
```

```
Out[56]: {0: 4, 1: 3, 2: 2, 3: 1, 4: 0}
```

Default values

if key in some_dict: value = some_dict[key] else: value = default_value

value = some_dict.get(key, default_value)

```
In [57]: words = ['apple', 'bat', 'bar', 'atom', 'book']  
by_letter = {}  
for word in words:  
    letter = word[0]  
    if letter not in by_letter:  
        by_letter[letter] = [word]  
    else:  
        by_letter[letter].append(word)  
by_letter
```

```
Out[57]: {'a': ['apple', 'atom'], 'b': ['bat', 'bar', 'book']}
```

세트(Set)

- **Set: 수학에서의 세트와 유사**

- 모든 원소값은 유일하다.
- 세트는 정렬되어 있지 않다
- 원소들은 서로 다른 데이터 타입으로 가능하다.

```
>>> s1 = set([1,2,3])  
>>> s1 {1, 2, 3}
```

```
>>> s2 = set("Hello")  
>>> s2 {'e', 'H', 'l', 'o'}
```

Set 생성

- **set function:**

- 빈 세트를 만들기 위해서는 `set()`
- 비어 있지 않는 세트를 만들기 위해서, `set(argument)` 와 같이 호출
- 여기에서 `argument` 는 순환이 가능한 원소들
 - `argument` 는 리스트 스트링 튜플 가능
 - 만약 `argument` 가 문자열이라면, 하나씩 떨어트리려눔
 - › 문자열을 리스트로 하나씩 떨어트리려 만듦
 - `argument` 안에 중복되는 원소값이 있다면 하나만 보여줌

```
>>> s1 = set([1,2,3])
>>> s1
{1, 2, 3}
>>> s2 = set("Hello")
>>> s2
{'e', 'H', 'l', 'o'}
>>> s3 = set([1,2,3,3,4])
>>> s3
{1, 2, 3, 4}
```

세트의 교집합

- Intersection of two sets: 두 셋트에서 모두 공통으로 갖고 있는 원소 값을 찾아냄
- 두가지 방법:
 - intersection 함수 이용
 - Format: `set1.intersection(set2)`
 - & 연산자 이용
 - Format: `set1 & set2`
 - 결과는 모두 새로운 Set 로 넘겨줌

```
>>> s1 = set([1, 2, 3, 4, 5, 6])
>>> s2 = set([4, 5, 6, 7, 8, 9])
>>> s1 & s2
{4, 5, 6}
>>> s1.intersection(s2)
{4, 5, 6}
```

세트의 차집합

- Difference of two sets: 두 셋트 상에서 유일하게 갖고 있는 원소들을 넘겨줌
- 두가지 방법:
 - Difference 함수 이용
 - Format: `set1.difference(set2)`
 - - 연산자 이용
 - Format: `set1 - set2`

```
>>> s1 = set([1, 2, 3, 4, 5, 6])
>>> s2 = set([4, 5, 6, 7, 8, 9])
>>> s1-s2
{1, 2, 3}
>>> s2-s1
{8, 9, 7}
>>> s1.difference(s2)
{1, 2, 3}
>>> s2.difference(s1)
{8, 9, 7}
>>> s2.difference(s2)
set()
```


세트의 합집합 - 교집합

- Symmetric difference of two sets: 두세트의 합집합에서 교집합을 뺀 모든 원소값 :

- `symmetric_difference` 사용
 - Format: `set1.symmetric_difference(set2)`
- ^연산자
 - Format: `set1 ^ set2`

```
>>> s1 = set([1, 2, 3, 4, 5, 6])
>>> s2 = set([4, 5, 6, 7, 8, 9])
```

```
>>> s1^s2
{1, 2, 3, 7, 8, 9}
>>> s2^s1
{1, 2, 3, 7, 8, 9}
>>> s1.symmetric_difference(s2)
{1, 2, 3, 7, 8, 9}
```

세트(Set)

```
In [17]: set([2, 2, 2, 1, 3, 3])  
{2, 2, 2, 1, 3, 3}
```

```
Out[17]: {1, 2, 3}
```

```
In [18]: a = {1, 2, 3, 4, 5}  
b = {3, 4, 5, 6, 7, 8}
```

```
In [19]: a.union(b)  
a | b
```

```
Out[19]: {1, 2, 3, 4, 5, 6, 7, 8}
```

```
In [20]: a.intersection(b)  
a & b
```

```
Out[20]: {3, 4, 5}
```

```
In [21]: c = a.copy()  
c |= b  
c  
d = a.copy()  
d &= b  
d
```

```
Out[21]: {3, 4, 5}
```

```
In [22]: my_data = [1, 2, 3, 4]  
my_set = {tuple(my_data)}  
my_set
```

```
Out[22]: {(1, 2, 3, 4)}
```

```
In [23]: a_set = {1, 2, 3, 4, 5}  
{1, 2, 3}.issubset(a_set)  
a_set.issuperset({1, 2, 3})
```

```
Out[23]: True
```

```
In [24]: {1, 2, 3} == {3, 2, 1}
```

```
Out[24]: True
```

세트(S

```
In [25]: strings = ['a', 'as', 'bat', 'car', 'dove', 'python']  
[x.upper() for x in strings if len(x) > 2]
```

```
Out[25]: ['BAT', 'CAR', 'DOVE', 'PYTHON']
```

```
dict_comp = {
```

```
set_comp = {
```

```
In [26]: unique_lengths = {len(x) for x in strings}  
unique_lengths
```

```
Out[26]: {1, 2, 3, 4, 6}
```

```
In [27]: set(map(len, strings))
```

```
Out[27]: {1, 2, 3, 4, 6}
```

```
In [28]: loc_mapping = {val : index for index, val in enumerate(strings)}  
loc_mapping
```

```
Out[28]: {'a': 0, 'as': 1, 'bat': 2, 'car': 3, 'dove': 4, 'python': 5}
```

| Python review3



명지대학교
MYONGJI UNIVERSITY

문자열 함수

Table 8-1 Some string testing methods

Method	Description
<code>isalnum()</code>	Returns true if the string contains only alphabetic letters or digits and is at least one character in length. Returns false otherwise.
<code>isalpha()</code>	Returns true if the string contains only alphabetic letters and is at least one character in length. Returns false otherwise.
<code>isdigit()</code>	Returns true if the string contains only numeric digits and is at least one character in length. Returns false otherwise.
<code>islower()</code>	Returns true if all of the alphabetic letters in the string are lowercase, and the string contains at least one alphabetic letter. Returns false otherwise.
<code>isspace()</code>	Returns true if the string contains only whitespace characters and is at least one character in length. Returns false otherwise. (Whitespace characters are spaces, newlines (<code>\n</code>), and tabs (<code>\t</code>).
<code>isupper()</code>	Returns true if all of the alphabetic letters in the string are uppercase, and the string contains at least one alphabetic letter. Returns false otherwise.

문자열 함수

Table 8-2 String Modification Methods

Method	Description
<code>lower()</code>	Returns a copy of the string with all alphabetic letters converted to lowercase. Any character that is already lowercase, or is not an alphabetic letter, is unchanged.
<code>lstrip()</code>	Returns a copy of the string with all leading whitespace characters removed. Leading whitespace characters are spaces, newlines (<code>\n</code>), and tabs (<code>\t</code>) that appear at the beginning of the string.
<code>lstrip(char)</code>	The <i>char</i> argument is a string containing a character. Returns a copy of the string with all instances of <i>char</i> that appear at the beginning of the string removed.
<code>rstrip()</code>	Returns a copy of the string with all trailing whitespace characters removed. Trailing whitespace characters are spaces, newlines (<code>\n</code>), and tabs (<code>\t</code>) that appear at the end of the string.
<code>rstrip(char)</code>	The <i>char</i> argument is a string containing a character. The method returns a copy of the string with all instances of <i>char</i> that appear at the end of the string removed.
<code>strip()</code>	Returns a copy of the string with all leading and trailing whitespace characters removed.
<code>strip(char)</code>	Returns a copy of the string with all instances of <i>char</i> that appear at the beginning and the end of the string removed.
<code>upper()</code>	Returns a copy of the string with all alphabetic letters converted to uppercase. Any character that is already uppercase, or is not an alphabetic letter, is unchanged.

문자열 함수

Table 8-3 Search and replace methods

Method	Description
<code>endswith(<i>substring</i>)</code>	The <i>substring</i> argument is a string. The method returns true if the string ends with <i>substring</i> .
<code>find(<i>substring</i>)</code>	The <i>substring</i> argument is a string. The method returns the lowest index in the string where <i>substring</i> is found. If <i>substring</i> is not found, the method returns -1.
<code>replace(<i>old</i>, <i>new</i>)</code>	The <i>old</i> and <i>new</i> arguments are both strings. The method returns a copy of the string with all instances of <i>old</i> replaced by <i>new</i> .
<code>startswith(<i>substring</i>)</code>	The <i>substring</i> argument is a string. The method returns true if the string starts with <i>substring</i> .

함수 정의와 함수 호출

함수는 이름을 갖고 있다.

- 함수의 이름 규칙:
 - Keyword를 함수 이름으로 사용 불가
 - 스페이스 사용 불가
 - 문자나 _(underscore)로 시작 가능
 - 함수 이름의 중간에는 문자, 숫자, _ 가능
 - 대소문자 구별

함수 이름은 함수에 의해 특별히 수행되는 어떤 결과를 주로 나타낸다.

- 함수의 이름에 때로는 동사 포함

함수 정의

```
def function_name():  
    statement  
    statement
```


함수 호출 예

```
# This program has two functions. First we
# define the main function.
def main():
    print('I have a message for you.')
    message()
    print('Goodbye!')

# Next we define the message function.
def message():
    print('I am Arthur')
    print('King of the Britons.')

# Call the main function.
main()
```

```
I have a message for you.
I am Arthur
King of the Britons.
Goodbye!
```

함수 - 들여 쓰기

- 모든 블록 들여쓰기 해야 한다

- 같은 블록은 같은 수의 스페이스가 필요
 - 탭이나 스페이스를 들여쓰기에 사용
 - 하지만 둘다 혼용하여 쓰는 것은 interpreter에게 혼돈을 줄 수 있음(한가지만!!)
 - IDLE 는 자동으로 들여 쓰기 기능
- 블록에서 빈 라인은 무시 가능

같은
스페이스 만큼
들여쓰기

```
def number_cal(a, b):  
    a_plub_b = a + b  
    a_mult_b = a * b  
    a_minums_b = a - b  
    a_divide_b = a / b  
    // 빈 라인은 무시가능  
    return a_plub_b, a_mult_b, a_minums_b, a_divide_b
```

함수 - 스코프

Namespaces, Scope, and Local Functions

```
In [33]: a = None
def bind_a_variable():
    global a
    a = []
bind_a_variable()
print(a)

[]
```

여러 변수 값을 넘길 경우

Python 에서 여러 변수 값을 넘겨야 할 경우에는 `return` 뒤에 , 를 연결하여 여러 값을 넘길 수 있다.

- Format: `return expression1, expression2, etc.`
- 이렇게 여러 변수넘기는 함수를 호출 시에는 넘겨주는 변수의 개수와 같은 개수만큼 = 연산자 왼쪽에 변수 할당

```
Height , weight, BMI = getWeightHeightBMI()
```

```
.....
```

```
def getWeightHeightBMI:
```

```
.....
```

```
.....
```

```
    return height, weight, bmi
```

여러 변수 값을 넘길 경우

Returning Multiple Values

```
In [54]: def f():  
         a = 5  
         b = 6  
         c = 7  
         return a, b, c  
  
a, b, c = f()
```

```
In [57]: return_value = f()  
         print(return_value)  
  
(5, 6, 7)
```

함수

```
In [34]: states = [' Alabama ', 'Georgia!', 'Georgia', 'georgia', 'Fl0rIda',  
                  'south carolina##', 'West virginia?']
```

```
In [35]: import re  
  
def clean_strings(strings):  
    result = []  
    for value in strings:  
        value = value.strip()  
        value = re.sub('[!#?]', '', value)  
        value = value.title()  
        result.append(value)  
    return result
```

```
In [36]: clean_strings(states)
```

```
Out[36]: ['Alabama',  
          'Georgia',  
          'Georgia',  
          'Georgia',  
          'Georgia',  
          'Florida',  
          'South Carolina',  
          'West Virginia']
```

함수

```
In [37]: def remove_punctuation(value):  
         return re.sub('[!#?]', '', value)  
  
         clean_ops = [str.strip, remove_punctuation, str.title]  
  
         def clean_strings(strings, ops):  
             result = []  
             for value in strings:  
                 for function in ops:  
                     value = function(value)  
                 result.append(value)  
             return result
```

```
In [38]: clean_strings(states, clean_ops)
```

```
Out[38]: ['Alabama',  
          'Georgia',  
          'Georgia',  
          'Georgia',  
          'Florida',  
          'South Carolina',  
          'West Virginia']
```

```
In [39]: for x in map(remove_punctuation, states):  
         print(x)
```

```
Alabama  
Georgia  
Georgia  
georgia  
FlOrIda  
south carolina  
West virginia
```

익명함수

Anonymous (Lambda) Functions

```
In [61]: def short_function(x):  
         return x * 2  
  
equiv_anon = lambda x: x * 2  
print(equiv_anon(2))  
  
4
```

```
In [59]: def apply_to_list(some_list, f):  
         return [f(x) for x in some_list]  
  
ints = [4, 0, 1, 5, 6]  
apply_to_list(ints, lambda x: x * 2)  
  
Out[59]: [8, 0, 2, 10, 12]
```

```
In [40]: strings = ['foo', 'card', 'bar', 'aaaa', 'abab']
```

```
In [41]: strings.sort(key=lambda x: len(set(list(x))))  
strings
```

```
Out[41]: ['aaaa', 'foo', 'abab', 'bar', 'card']
```


부분 함수

Currying: Partial Argument Application

```
In [63]: def add_numbers(x, y):  
         return x + y
```

```
In [64]: add_five = lambda y: add_numbers(5, y)  
         print(add_five(10))
```

15

이터레이터를 생성해주는 함수

Generators :이터레이터를 생성해주는 함수

```
In [42]: some_dict = {'a': 1, 'b': 2, 'c': 3}
         for key in some_dict:
             print(key)
```

```
a
b
c
```

```
In [43]: dict_iterator = iter(some_dict)
         dict_iterator
```

```
Out[43]: <dict_keyiterator at 0x10f15ab88>
```

```
In [44]: list(dict_iterator)
```

```
Out[44]: ['a', 'b', 'c']
```

```
In [86]: dict_iterator = iter(some_dict.values())
         dict_iterator
```

```
Out[86]: <dict_valueiterator at 0x10f1903b8>
```

```
In [87]: list(dict_iterator)
```

```
Out[87]: [1, 2, 3]
```

클래스 정의

- **Class definition(클래스 정의):** methods and data attributes와 로 정의
- **Format: begin with `class Class_name:`**
 - 클래스 이름은 주로 대문자로 시작
 - Method정의는 Python 내의 함수만드는 것과 동일
 - self parameter: 클래스안에 모든 메소드들에 존재 - 현재 동작하고 있는 객체를 지칭

클래스 정의 (cont'd.)

- **Initializer method(초기 메소드)**: 클래스에서 **instance**가 한개 생성될때 자동으로 생성
 - 객체의 data attributes 값을 초기화 하고 객체 생성을 위해 `self` 파라미터값을 정의
 - Format: `def __init__ (self):`
 - 보통 클래스 내의 첫 메소드

클래스 정의(cont'd.)

- 클래스 새로운 **instance** 를 생성하기 위하여 클래스내 **initializer method**를 호출
 - Format: `My_instance = Class_Name()`
- 클래스 내의 특별한 **method**를 호출 하기 위하여 **.** 을 사용
 - Format: `My_instance.method()`
 - Because the `self` parameter references the specific instance of the object, the method will affect this instance
 - Reference to `self` is passed automatically

Circle.py

```
import math

class Circle:
    # Construct a circle object
    def __init__(self, radius = 1):
        self.__radius = radius

    def getRadius(self):
        return self.__radius

    def getPerimeter(self):
        return 2 * self.__radius * math.pi

    def getArea(self):
        return self.__radius * self.__radius * math.pi

    def setRadius(self, radius):
        if radius >= 0:
            self.__radius = radius

>>> import circle
>>> c = Circle(5)
>>> c.__radius
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    c.__radius
AttributeError: 'Circle' object has no attribute '__radius'
>>> c.getRadius()
5
>>> c.setRadius(10)
>>> c.getRadius()
10
>>> |
```