AI 프로그래밍 4

융합학과 권오영
oykwon@koreatech.ac.kr

KOREATECH
한국기술교육대학교

복합 자료형 (Built-in Collection Data Types)

복합 데이터 타입

❖ Python에 사용하는 복합 자료형

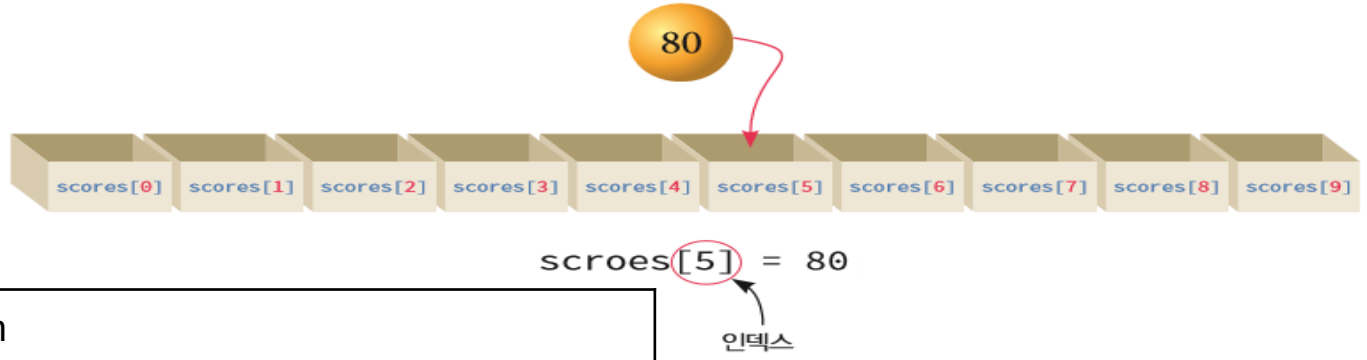
- 리스트: an ordered collection of zero or more references to data objects
- 튜플 : an ordered collection of zero or more references to data objects but immutable
- 집합: an unordered collection of zero or more immutable data objects
- 사전: collections of associated pairs of items where each pair consists of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

리스트 (list)	튜플 (tuple)	리스트 (list)
['aaa', 123] 1 dimensional sequence of different data type objects Can be updated	('aaa', 123) 1 dimensional sequence of different data type objects Can not be updated	{'no': 123, 'name':'anonymous' } hash table type associative array key-value pairs

리스트

❖ list : an ordered sequence of value.

- Syntax -> [elem*1*, elem*2*, … , elem*n*]
- 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 (an ordered 이므로)

예) [1,2,3,4][1:3][1] → 3

[1,2,3,4][1:3] → [2,3,4], [2,3,4][1] → 3

[1,2,3,4][-1] → 4

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

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 = list()
L3.append(3)

L4 = list(range(10, 0 ,-1))
L5 = list('Hello')
```

```
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
mylist[2]	인덱스 2에 있는 요소
mylist[2] = 3	인덱스 2에 있는 요소를 3으로 설정한다.
del mylist[2]	인덱스 2에 있는 요소를 삭제한다.
len(mylist)	mylist의 길이를 반환한다.
"value" in mylist	"value"가 mylist에 있으면 True
"value" not in mylist	"value"가 mylist에 없으면 True
mylist.sort()	mylist를 정렬한다.
mylist.index("value")	"value"가 발견된 위치를 반환한다.
mylist.append("value")	리스트의 끝에 "value"요소를 추가한다.
mylist.remove("value")	mylist에서 "value"가 나타나는 위치를 찾아서 삭제한다.

List Comprehension

❖ 리스트를 빠르게 만들고 변경할 수 있는 방법

- 수학의 조건식과 유사함
- 문법

  [expression + context]

  expression: 리스트에 들어갈 원소에 수행할 일을 명시

  context: 어떤 원소를 선택할지 기술한 부분으로 for 와 if 문이 사용

- 예시

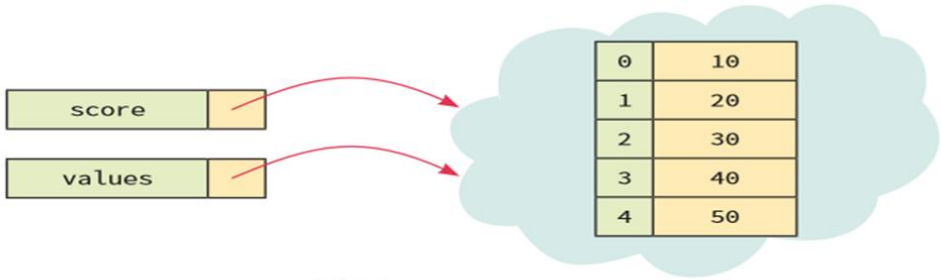
  [n * 5 for n in range(10)]

  [n for n in range(10) if n%2==0]

리스트 복사

❖ 얕은 복사

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



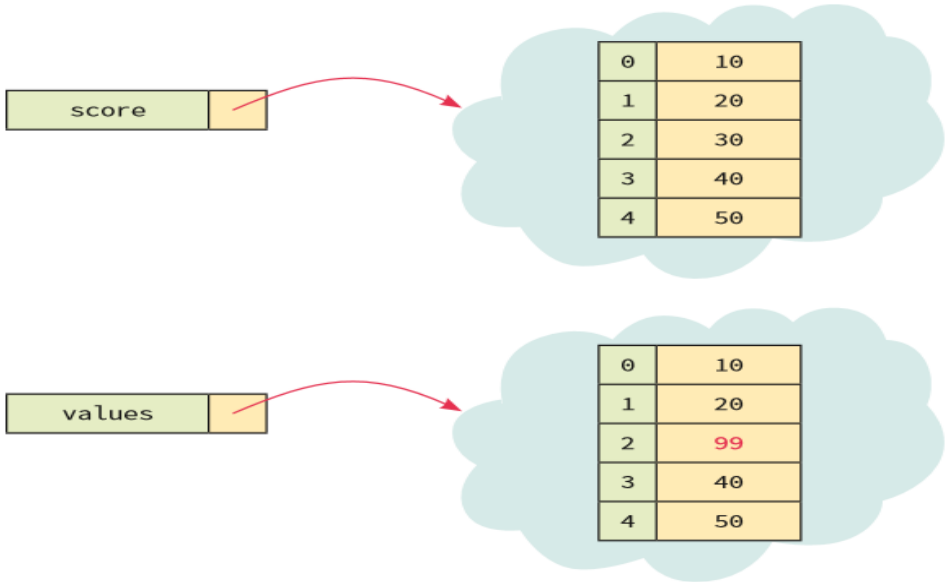
참조

❖ 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]
```

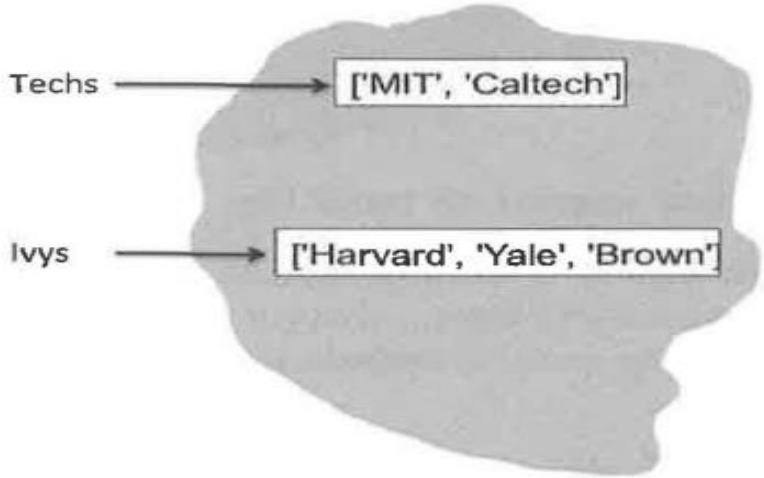


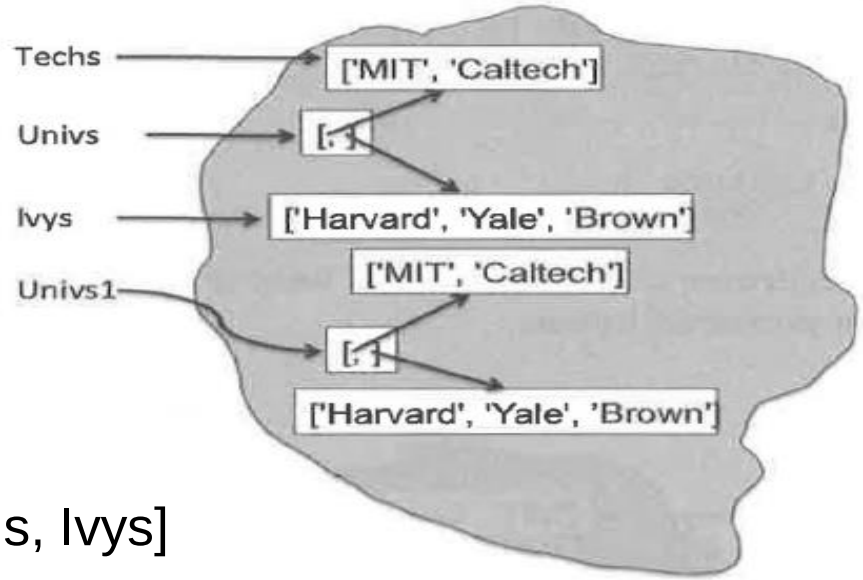
Mutability

❖ 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’]

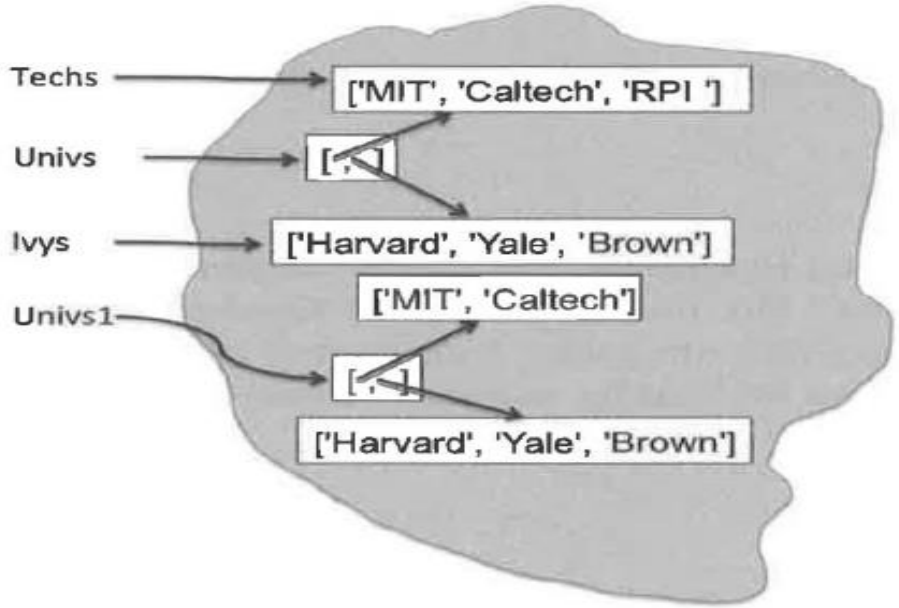




❖ Univs = [Techs, Ivys]
Univs1 = [[‘MIT’, ‘Caltech’], [‘Harvard’, ‘Yale’, ‘Brown’]]

Mutability

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



❖ Univs를 출력하게 되면 RPI가 추가된 Techs를 참조해서 리스트를 출력한다.
(aliasing발생) → 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’

KOREATECH 한국기술교육대학교

스트링

❖ Type checking: type 간의 연산이 가능한지 조사

str*str : 오류

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

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

```
>>> 'abc'[0]
>>> 'abc'[3]
>>> 'abc'[-1]
```

a	b	c
0	1	2
-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'
```

튜플

❖ 튜플(Tuple) : ordered sequences of elements

- Syntax -> (elem*1*, elem*2*, … , elem*n*)
- 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	'', 'a', 'abc'	No
tuple	any type	(), (3,), ('abc', 4)	No
list	any type	[], [3], ['abc', 4]	Yes

※ 계산도중 내용을 추가할 수 있는 list가 많이 사용된다.

집합

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

- Syntax -> { elem*1*, elem*2*, … , elem*n* } // 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	a_set.union(other_set)	Returns a new set with all elements from both sets
intersection	a_set.intersection(other_set)	Returns a new set with only those elements common to both sets
difference	a_set.difference(other_set)	Returns a new set with all items from first set not in second
issubset	a_set.issubset(othe_rset)	Asks whether all elements of one set are in the other
add	a_set.add(item)	Adds item to the set
remove	a_set.remove(item)	Removes item from the set
pop	a_set.pop()	Removes an arbitrary element from the set
clear	a_set.clear()	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	a_dict.keys()	Returns the keys of the dictionary in a dict_keys object
values	a_dict.values()	Returns the values of the dictionary in a dict_values object
items	a_dict.items()	Returns the key-value pairs in a dict_items object
get	a_dict.get(k)	Returns the value associated with k, None otherwise
get	a_dict.get(k, alt)	Returns the value associated with k, alt otherwise

함수

함수

- 프로그래밍에서 함수란?
- 일정한 작업을 수행하기 위한 일련의 코드 조각


```
sum = 0;
for i in range(1, 11)
    sum += i;
```

```
sum = 0;
for i in range(1, 21)
    sum += i;
```

- 함수이름은 함수의 목적을 설명하는 동사 또는 동사+명사를 사용

```
get_sum(1, 10)
get_sum(1, 20)
```

```
def get_sum(start, end)
    sum = 0;
    for i in range(start, end+1)
        sum += i;
    return sum
```

KOREATECH 한국기술교육대학교

함수 사용의 장점

- 함수를 작성하면, 어디서든 재사용할 수 있다.
- Information Hiding: 구현의 자세한 사항을 사용자로부터 숨길 수 있다. (추상화)
- 복잡도 감소
  - 복잡한 프로그래밍 작업을 더 간단한 작업들로 분해할 수 있다.
  - 가독성이 증대되고, 유지 관리도 쉬워진다.

- 함수 정의

[return type] name of function (list of formal parameters) {

body of function (statements…)

[return value]

}

[ ]로 둘러싸인 부분은 선택사항

함수

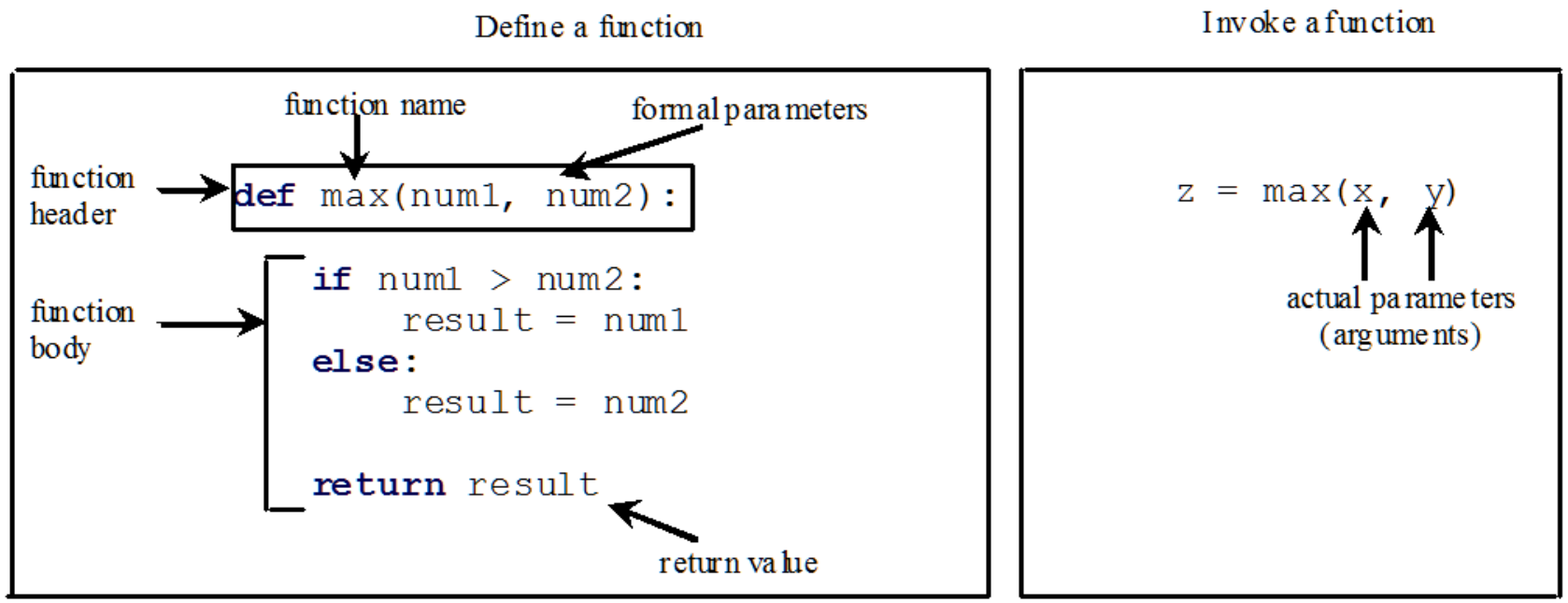
- Python에서 함수란?
- 일정한 작업을 수행하기 위한 일련의 코드 조각
  - 함수는 이름을 가지고 있고, 함수의 작업을 수행하기 위하여 코드 내에 함수의 이름을 명시한다. 이것을 함수 calling 이라 한다.
  - 함수는 하나 이상의 입력값이 필요하다. 입력값이 없을 수도 있다.
  - 함수는 하나 값을 되돌려줄 수 있고, 그렇지 않을 수도 있다. 함수를 값을 되돌려주는 것을 returning이라 한다.

```
def my_function():
    print("Hello from a function")

my_function()
```

함수 사용의 장점

- 함수를 작성하면, 어디서든 재사용할 수 있다.
- Information Hiding: 구현의 자세한 사항을 사용자로부터 숨길 수 있다. (추상화)
- 복잡도 감소
  - 복잡한 프로그래밍 작업을 더 간단한 작업들로 분해할 수 있다.
  - 가독성이 증대되고, 유지 관리도 쉬워진다.



함수

❖ 함수 정의

def *name of function* (*list of formal parameters*):
    body of function

```
def max(x, y):
    if x>y:
        return x
    else
        return y

z = max(3,4)
    # 3,4 actual parameters (or arguments)
```

❖ 함수의 파라미터들 매칭

- Positional -> 각 해당 위치의 파라미터들로 매칭, 즉 첫 actual parameter는 첫 formal parameter와 매칭
- Keyword arguments -> 위치와 상관없이 formal parameter 이름을 사용해서 actual parameter를 할당

함수파라미터

❖ def printName(firstName, lastName, reverse):

```
    if reverse:
        print (lastName + ', ' + firstName)
    else:
        print (firstName, lastName)
```

❖ 아래 호출이 모두 동일함

```
printName('Olga', 'Puchmajerova', False)
printName('Olga', 'Puchmajerova', reverse = False)
printName('Olga', lastName = 'Puchmajerova', reverse = False)
printName(lastName = 'Puchmajerova', firstName = 'Olga', reverse = False)
```

❖ Keyword argument 뒤에 non-keyword argument가 오는 것은 오류

```
printName('Olga', lastName = 'Puchmajerova', False)
```

default 파라미터

❖ def printName(firstName, lastName, reverse = False):

```
    if reverse:
        print lastName + ', ' + firstName
    else:
        print firstName, lastName
```

※ python 2.x 은 print 에 ( )가 필요없고, python 3.x에서는 ( )가 필요하다.

❖ 함수파라미터중 reverse의 default 값을 False로 설정

```
printName('Olga', 'Puchmajerova')
printName('Olga', 'Puchmajerova', True)
printName('Olga', 'Puchmajerova', reverse = True)
```

가변 파라미터

❖ 파라미터의 수가 정해지지 않는 경우

```
def 함수이름 (매개변수, 매개변수, … , *가변매개변수):
    함수몸체
```

```
def print_n_times(n, *values):
        for i in range(n):
            for value in values:
                print(value)
            print()

print_n_times(3, “Hello”, “Fun”, “Python Programming”)
```