

# Python

List, tuple, dictionary, set, 그리고 For loop

# Dictionary

```
dictionary_type_variable =  
    { key1:value1, key2:value2, key3:value3 }
```

immutable한 키(key)와 mutable한 값(value)으로 매핑되어 있는 순서가 없는 집합.

```
>>> dictionary_type_variable = {}  
>>> dictionary_type_variable["addr"] = "Seoul"  
>>> print(dictionary_type_variable)  
>>> print(dictionary_type_variable["addr"])
```

```
>>> new_dict = {  
    "brand": "Honda",  
    "model": "Civic",  
    "year": 1995  
}  
>>> print(new_dict)  
>>> print(new_dict["model"])  
>>> new_dict["model"] = "No. 5"  
>>> print(new_dict)
```

```
>>> Customer_1= {  
    'username': 'john-sea',  
    'online': false,  
    'friends': 100  
}  
>>> 원하는 동작을 이것 저것 연습...
```

# Set

```
set_type_variable =  
{ value1, value2, value3 }
```

mutable한 집합(value)으로 순서가 없으며, 집합안에서는 unique한 집합을 갖는다.

```
>>> aSet = set()
```

```
>>>
```

```
set_type_variable =  
    { value1, value2, value3 }
```

```
>>> aSet = set()
```

```
>>> aSet.add(1)
```

```
>>> print(aSet)
```

```
>>> aSet.add(2)
```

```
>>> print(aSet)
```

```
>>> aSet.add(1)
```

```
>>> print(aSet)
```

```
>>> aSet = {} # 집합(set)을 만드는 방법이 아니다, dictionary가 만들어진다.
```

```
>>> print(type(aSet))
```

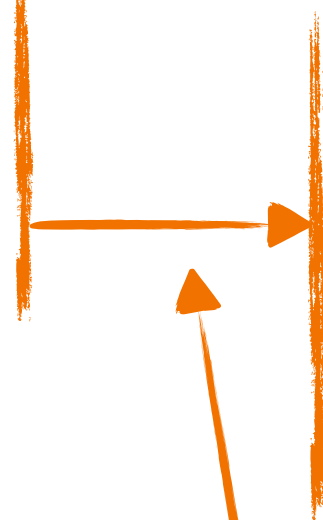
# For loop



For loop is a handy way for iterating over a sequence such as a list, tuple, dictionary, set, string, etc.

iterable data structure

```
for variable in iterable_data_structure:
```



들여쓰기 indentation

} Block

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

```
aSet = set(a)
```

```
for x in a:
```

```
    print(x, end=" ")
```

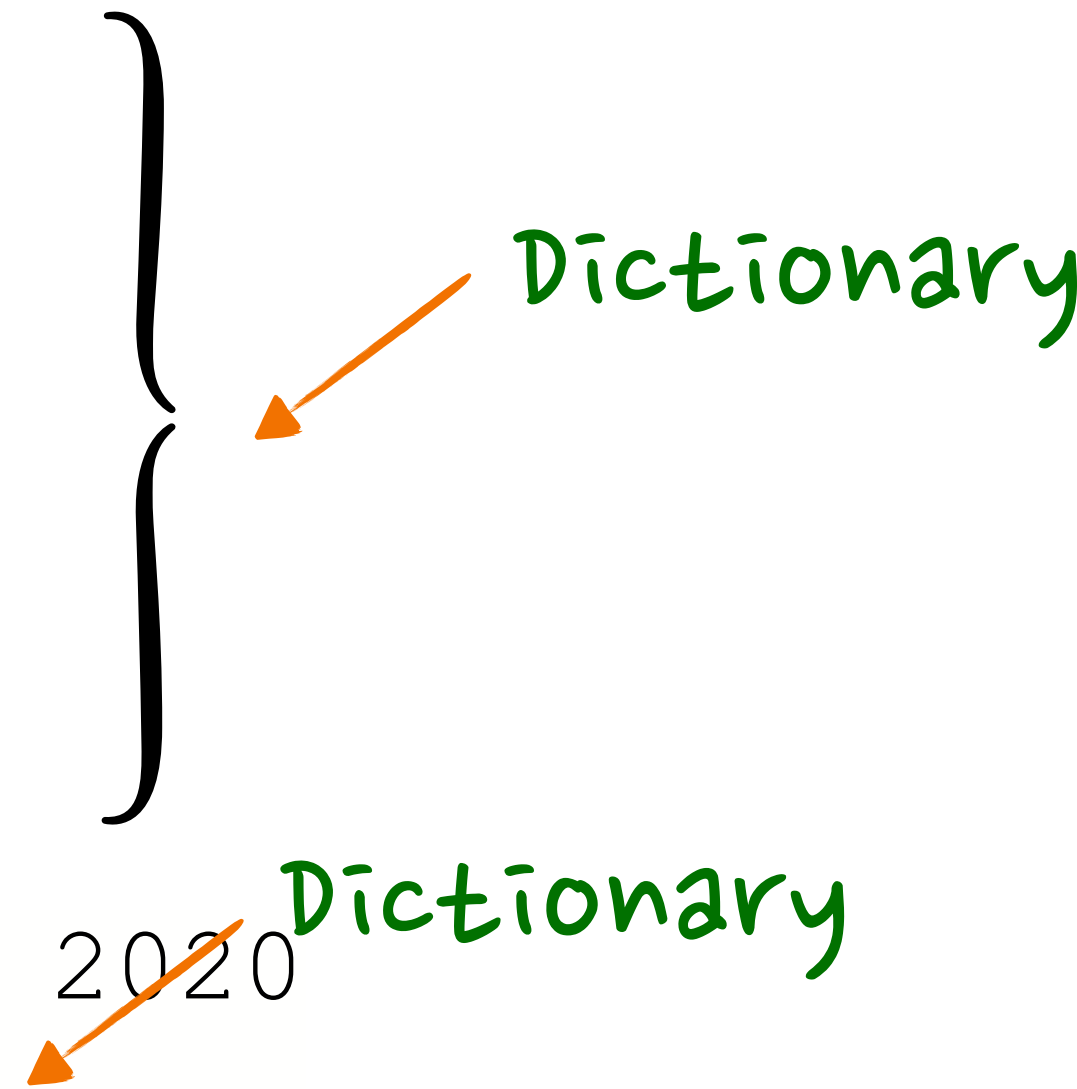
```
print("")
```

```
for x in aSet:
```

```
    print(x, end=" ")
```

```
print("")
```

```
new_dict= {  
    "brand": "Honda",  
    "model": "Civic",  
    "year": 1995  
}  
new_dict["year"] = 2020  
for key in new_dict:  
    print(key, new_dict[key])  
  
for key, value in new_dict.items():  
    print(key, value)
```



String

```
for x in "Hello World":  
    print(x, end = "")
```

range() 함수 : 연속된 숫자(정수)를 만들  
어주는 함수

```
fruit_list = []  
for x in range(1,4):  
    fruit_list += ["apple"]  
    print(fruit_list)
```

range() 함수 : 연속된 숫자(정수)를 만들어주는 함수

**range (stop)**

range (10) # 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

**range (start, stop)**

range (1, 11) # 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

**range (start, stop, step)**

range (0, 20, 2) # 0, 2, 4, 6, 8, 10, 12, 14, 16, 18

range (20, 0, -2) # 20, 18, 16, 14, 12, 10, 8, 6, 4, 2

range() 함수 : 연속된 숫자(정수)를 만들어주는 함수

**range (stop)**

range (10) # 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

```
aList = list(range(10))
```

```
print(aList)
```

**range (start, stop)**

range (1, 11) # 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

```
aList = list(range(1, 11))
```

```
print(aList)
```

**range (start, stop, step)**

range (0, 20, 2) # 0, 2, 4, 6, 8, 10, 12, 14, 16, 18

```
aList = list(range(0, 20, 2))
```

```
print(aList)
```

range (20, 0, -2) # 20, 18, 16, 14, 12, 10, 8, 6, 4, 2

```
aList = list(range(20, 0, -2))
```

```
print(aList)
```

“

aList = [ 23, 56, 12, 7, 9, 201, 33]과 같은 list가 주어졌을 때, list안에 최대값을 찾아 해당 index를 출력하시오.

”

“

철수는 2000년 11월 3일에 태어났다.  
오늘(2020년 11월 21일)은 태어난지 20개월째 되는 날일까?

”



```
daysOfMonth = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
```

```
days = 0
```

```
days += daysOfMonth[monthOfBirthday-1]
```

```
if (monthOfBirthday == 2 ) :
```

```
    if (yearOfBirthday % 4 == 0 and  
        yearOfBirthday % 100 != 0 or  
        yearOfBirthday % 400 == 0 ) :
```

```
        days = days + 1
```

```
days -= dayOfBirthday
```

```
currentMonth = monthOfBirthday + 1
while (currentMonth <= 12) :
    days += daysOfMonth[currentMonth-1]
    if (currentMonth == 2) :
        if (yearOfBirthday % 4 == 0 and
            yearOfBirthday % 100 != 0 or
            yearOfBirthday % 400 == 0 ) :
            days = days + 1

currentMonth+=1
```

```
yearOfBirthday += 1
while (yearOfBirthday < yearOfToday) :
    days = days + 365
    if (yearOfBirthday % 4 == 0 and
        yearOfBirthday % 100 != 0 or
        yearOfBirthday % 400 == 0 ) :
        days = days + 1

yearOfBirthday += 1
```

```
currentMonth = 1
while (currentMonth < monthOfToday) :
    days += daysOfMonth[currentMonth-1]
    if (currentMonth == 2) :
        if (yearOfBirthday % 4 == 0 and
            yearOfBirthday % 100 != 0 or
            yearOfBirthday % 400 == 0 ) :
            days = days + 1

    currentMonth+=1
days += dayOfToday
print(days, "days")
```

“

철수가 4월 태어났다면?

”

“

주어진 배열  $nums = [2, 7, 11, 15]$ 가 있을 때 배열 안의  
두 개의 원소를 더해 주어진  $target = 9$ 에 만족할 때, 해당  
인덱스  $[0, 1]$ 을 보여주는 프로그램을 작성하라.

”

# Function

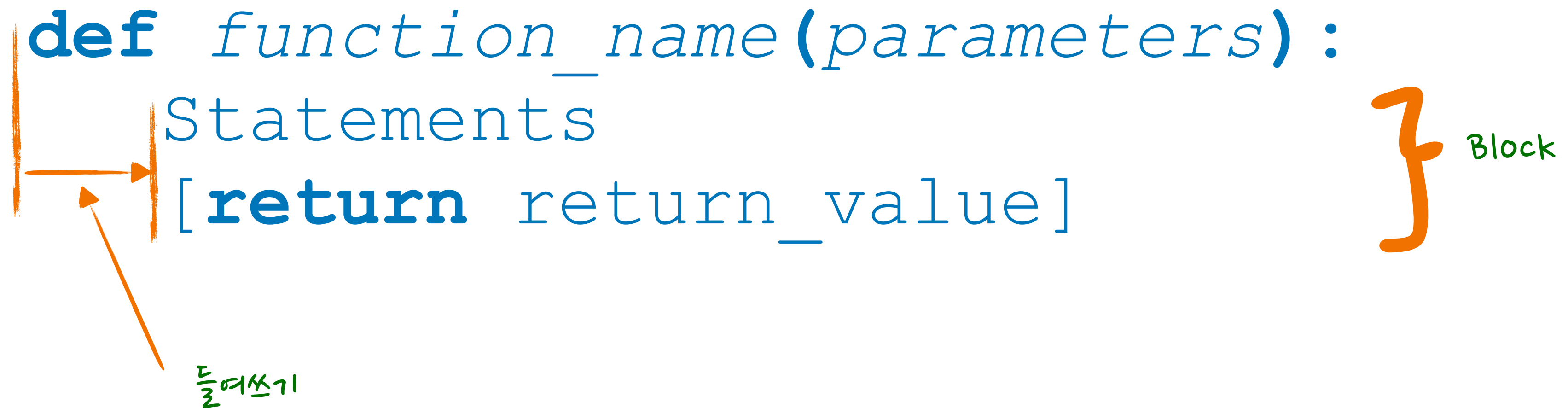
# Function

- Built-in Functions
- Define a Function
- lamda



# Function

```
def function_name(parameters):  
    Statements  
    [return return_value]
```



The diagram illustrates the components of a function definition. A vertical orange line marks the start of the function. A horizontal orange arrow points from this line to the word 'Statements', which is also underlined. Another orange arrow points from the word 'return' to the handwritten Korean text '돌려쓰기' (to return/reuse). To the right of the function body, a large orange closing curly brace '}' is shown, with the word 'Block' written in green next to it.

```
def IsLeapYear(year):  
    result = (year % 4 == 0 and year % 100 != 0 or year %  
400 == 0)  
    return result
```

# Lambda

```
[lambda_name = ] lambda parameters : expression
```

```
lambda_name(parameters)
```

```
isLeapYear = lambda x : (x % 4 == 0 and x % 100 != 0 or x % 400  
== 0) and True or False  
print(isLeapYear(2020))
```

```
[variable_name = ] (lambda parameters : expression)
```

```
(parameter_values)
```

```
print((lambda x : (x % 4 == 0 and x % 100 != 0 or x % 400 == 0)  
and True or False)(2020))
```

# Q & A