클라우드 컴퓨팅과 AI서비스 (파이썬)

융합학과 권오영

oykwon@koreatech.ac.kr

KOREATECH 한국기술교육대학교

학습내용

❖ 파이썬 언어 소개

- 함수
- 객체
- 파일과 모듈
- 예외처리

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;
```

```
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
```

- 함수이름은 함수의 목적을 설명하는 동사 또는 동사+명사를 사용

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

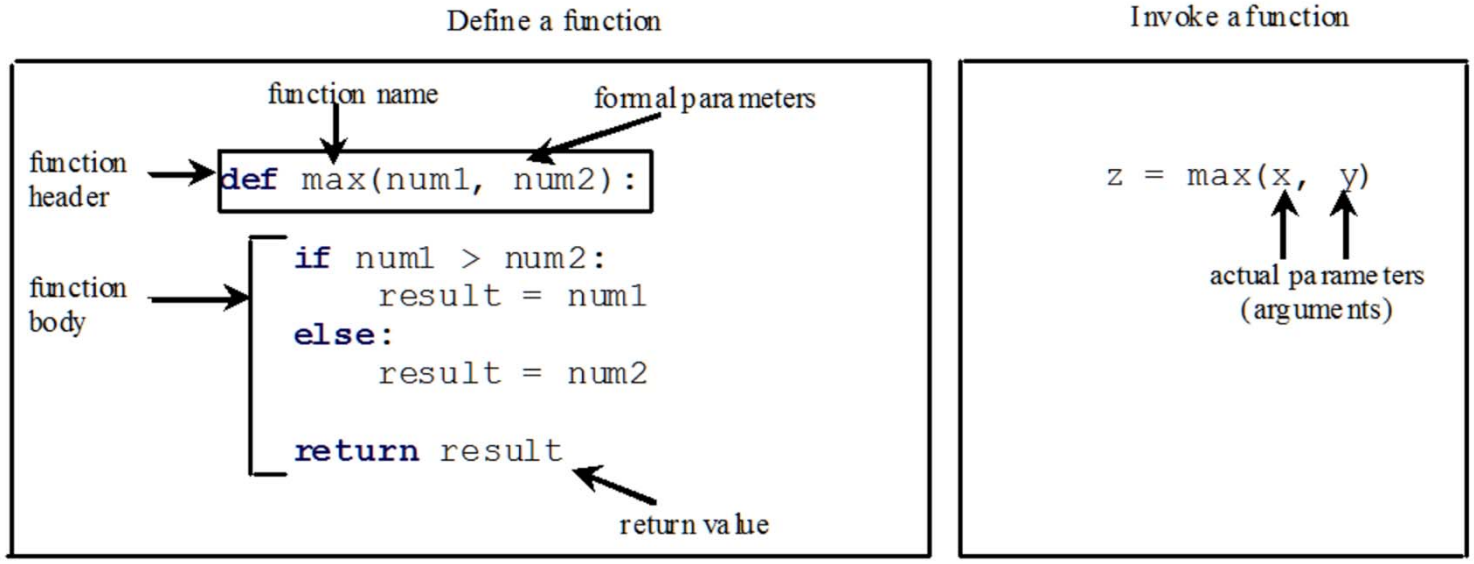
함수

- 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

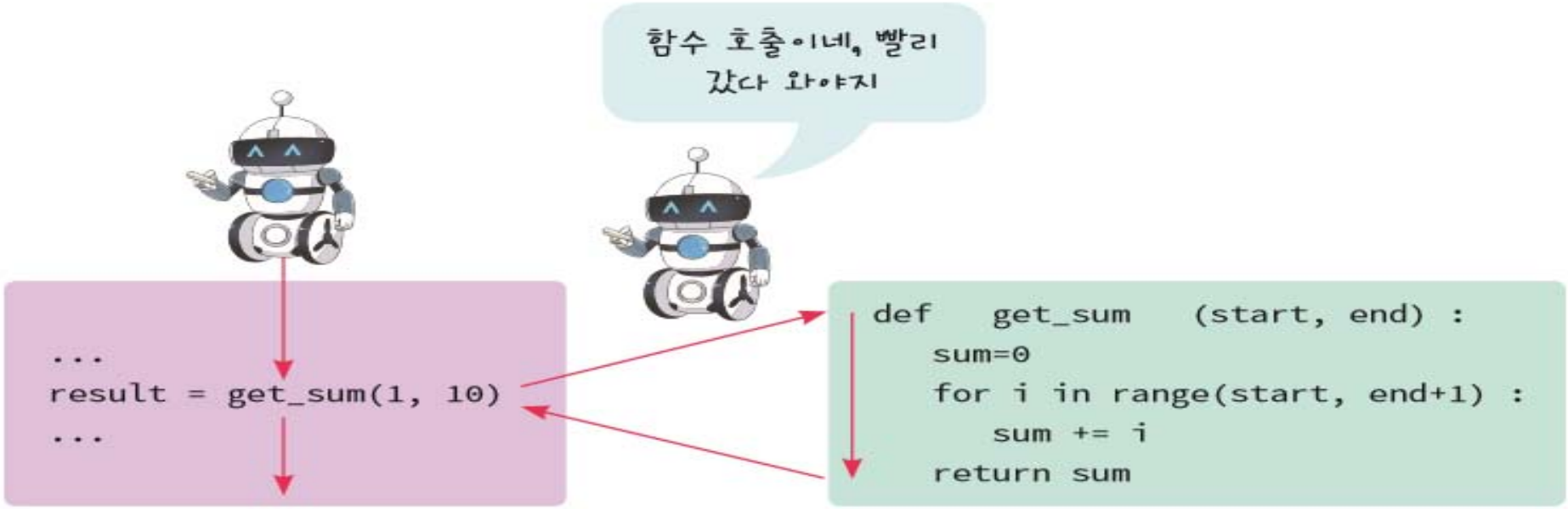
❖ 파라미터의 수가 정해지지 않는 경우

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”)
```

함수호출 흐름

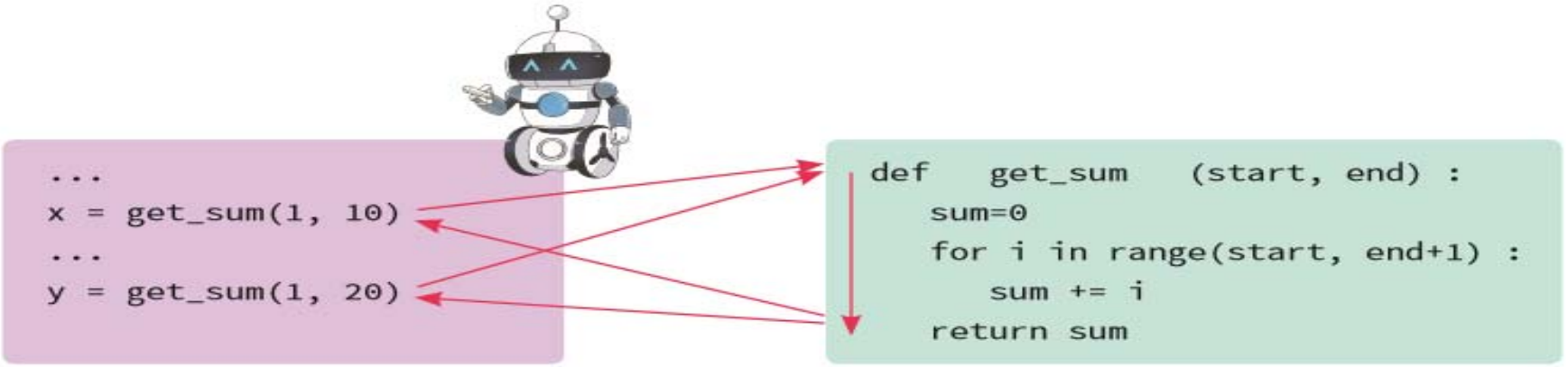
❖ 함수 호출을 통한 재사용
(돌아올 주소를 스택에 저장)



KOREATECH 한국기술교육대학교

함수호출 흐름

❖ 함수 호출을 통한 재사용
(돌아올 주소를 스택에 저장)



범위(Scoping)

❖ 함수는 자신의 name space (scope)을 형성

```
def f(x): #name x used as formal parameter
    y = 1
    x = x + y
    print ('x =', x)
    return x

x = 3
y = 2
z = f(x)
print ('z =', z)
print ('x =', x)
print ('y =', y)
```

- 수행결과는

```
x = 4  # 함수 내부 변수
z = 4
x = 3  # top level 변수
y = 2
```

범위(Scoping)

❖ 정적 범위 (lexical scoping) -> 프로그램의 정적인 내포관계에 따라 변수의 영향을 끼치는 범위가 결정되는 방법

```
def f(x):
    def g():
        x = 'abc'
        print ('x =', x)
    def h():
        z = x
        print ('z =', z)
    x = x + 1
    print ('x =', x)
    h()
    g()
    print ('x =', x)
    return g

x = 3
z = f(x)
print ('x =', x)
print ('z =', z)
z()
```

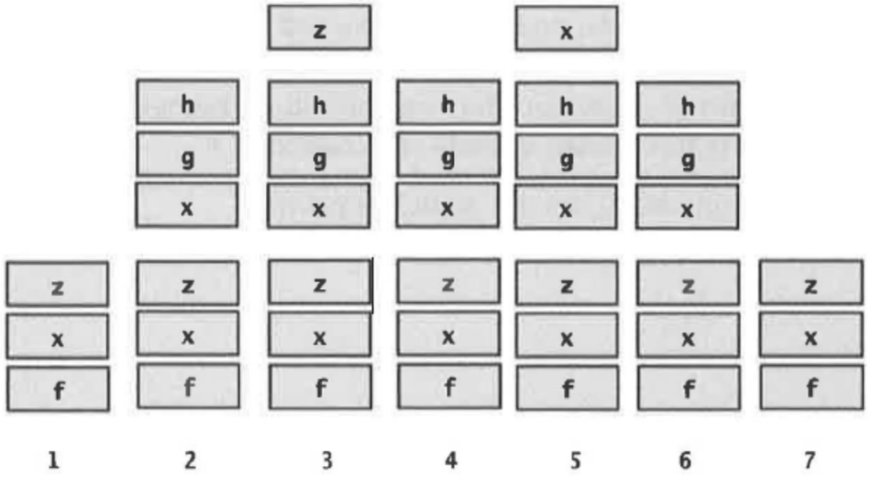


Figure 4.4 Stack frames

실행

x = 4
z = 4
x = abc
x = 4
x = 3
z = <function g at 0x… >
x = abc

Scope

❖ 다른 예제

```
def f():
    print(x)
def g():
    print(x)
    x = 1


x = 3
f()
x = 3
g()
```

❖ 함수 f()는 정상작동하나 g()는 오류
→ g 내부에서 x를 정의하는 부분이 있어서 x를 local로 간주

❖ Global(전역)과 local(지역)변수를 잘 구분해야함

전역변수

❖ 전역변수(global variable)

- 가급적 전역변수의 사용은 최소화 하는 것이 바람직하다.
- 파이썬에서는 global 이라는 한정자를 변수 앞에 붙여서 전역변수임을 알려준다.

```
def fib(x):
    """Assumes x an int >= 0
       Returns Fibonacci of x"""
    global numFibCalls
    numFibCalls += 1
    if x == 0 or x == 1:
        return 1
    else:
        return fib(x-1) + fib(x-2)

def testFib(n):
    for i in range(n+1):
        global numFibCalls
        numFibCalls = 0
        print 'fib of', i, '=', fib(i)
        print 'fib called', numFibCalls, 'times.'
```

함수 명세(specification)

- 함수 명세 -> 사용자에게 함수에서 가정한 것과 결과 값을 설명해준다.
  - Assumptions : 함수를 사용하는 사람에게 제약조건을 명확히 알려주는 역할을 한다.
    findRoot 함수에서 Assumes ~ power >= 1 부분이다.
  - Guarantees : 가정에 맞게 함수를 호출하면, 함수가 제공해야하는 조건을 기술한다.
    findRoot 함수의 Returns ~ returns None 부분이다.
- 함수를 선언할 때 위의 두가지를 꼭 기술하자. 최소한 formal parameters들에 대한 설명(Assumptions)과 함수의 반환값(Guarantees)에 대한 설명은 꼭 하는 습관을 갖도록 노력하자.
- 함수는 프로그램을 작성하는 elements
  - Decomposition : 문제를 작은 모듈들로 분해해서 구조를 만들어 낸다.
  - Abstraction : 자세한 처리 과정은 함수의 몸체 안으로 숨기는 역할을 한다.
- 추상화(Abstraction)
  - 프로그래머는 함수 명세를 보고 코드를 구현
    -> 일을 규모있게 처리할 수 있다.

Help 함수

```
def findRoot(x, power, epsilon):
    """Assumes x and epsilon int or float, power an int,
           epsilon > 0 & power >= 1
       Returns float y such that y**power is within epsilon of x.
           If such a float does not exist, it returns None"""
    if x < 0 and power%2 == 0:
        return None
    low = min(-1.0, x)
    high = max(1.0, x)
    ans = (high + low)/2.0
    while abs(ans**power - x) >= epsilon:
        if ans**power < x:
            low = ans
        else:
            high = ans
        ans = (high + low)/2.0
    return ans

def testFindRoot():
    epsilon = 0.0001
    for x in (0.25, -0.25, 2, -2, 8, -8):
        for power in range(1, 4):
            print 'Testing x = ' + str(x) +\
                  ' and power = ' + str(power)
            result = findRoot(x, power, epsilon)
            if result == None:
                print '   No root'
            else:
                print '   ', result**power, '~=', x
```

- """ doc string """ : 다중라인을 포함하고,
  help 함수는 doc string을 보여준다.
  help(findRoot) 하면
  Assumes x and … 라는 doc string을 확인할 수 있다.
  -> 직접 함수를 작성 확인

- 함수 findRoot와 findRoot가 올바로 작동하는지 검증하는 testFindRoot함수 작성
- 검증을 하는 testFindRoot함수를 작성하는 것이 시간을 낭비하는 것 처럼 생각(초보자)되지만 실제로는 큰 이득을 얻는 행동(숙련자)이다.
- 디버깅과정의 단축

Recursion

- 재귀함수 (함수 자신을 호출)
  - Base case
  - Recursive(inductive) case
- 수학적 귀납법을 생각
  - 초기조건 -> base case
  - 가정 (n) 을 충족
  - 다음 스텝(n+1)은? -> inductive case
- 팩토리얼 계산 (n!)
- 초기조건 1! = 1
- 가정 n! 을 구했다고 가정
- 다음 (n+1)! 은 ?

  (n+1)! = (n+1) * n!

Recursion

❖ 반복(iterative)법과 재귀(recursive)법

```
def factI(n):
    """Assumes that n is an int > 0
       Returns n!"""
    result = 1
    while n > 1:
        result = result * n
        n -= 1
    return result

def factR(n):
    """Assumes that n is an int > 0
       Returns n!"""
    if n == 1:
        return n
    else:
        return n*factR(n - 1)
```

Recursion(피보나치)

❖ 토기의 번식속도 : fib(0) = 1, fib(1) = 1, fib(n+2) = fib(n+1) + fib(n), n>=0

Month	Females
0	1
1	1
2	2
3	3
4	5
5	8
6	13

```
def fib(n):
    """Assumes n an int >= 0
       Returns Fibonacci of n"""
    if n == 0 or n == 1:
        return 1
    else:
        return fib(n-1) + fib(n-2)

def testFib(n):
    for i in range(n+1):
        print 'fib of', i, '=', fib(i)
```

Recursion(Palindromes)

❖ 좌우 대칭 형 문자열, doggod

```
def isPalindrome(s):
    """Assumes s is a str
       Returns True if the letters in s form a palindrome;
         False otherwise. Non-letters and capitalization are ignored."""

    def toChars(s):
        s = s.lower()
        letters = ''
        for c in s:
            if c in 'abcdefghijklmnopqrstuvwxyz':
                letters = letters + c
        return letters

    def isPal(s):
        if len(s) <= 1:
            return True
        else:
            return s[0] == s[-1] and isPal(s[1:-1])

    return isPal(toChars(s))
```

❖ 내부 함수

- toChars -> 문자를 모두 소문자로 바꾸고, non-letter는 제거 (s.lower(): s라는 스트링 객체가 가지고 있는 lower()라는 매소드 호출 (dot notation))
- isPal -> 재귀함수

객체

출처:https://runestone.academy/runestone/books/published/pythonds3/Introduction/ObjectOrientedProgramminginPythonDefiningClasses.html

객체

❖ 사전정의

1. 주체로부터 독립되어 있는 인간의 인식과 실천의 대상
2. 의사나 행위가 미치는 대상

❖ 컴퓨터 과학에서 **객체** 또는 **오브젝트**(object)는 클래스에서 정의한 것을 토대로 메모리(실제 저장공간)에 할당된 것으로 프로그램에서 식별자(변수)에 의해 참조되는 공간을 의미

- 객체는 클래스의 인스턴스
  (클래스는 틀(붕어빵틀), 객체는 틀을 바탕으로 만들어진 실체(붕어빵))

❖ 객체(object)는 어떠한 속성값과 행동을 가지고 있는 데이터

- 정수, 실수, 리스트등 각종 자료형도 모두 객체임

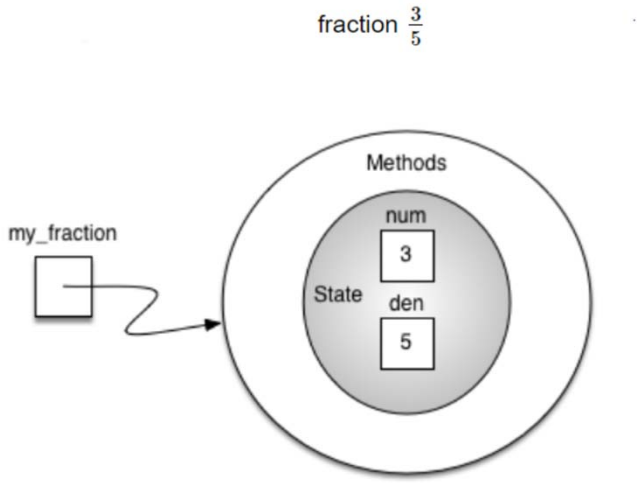
객체의 정의

❖ 클래스(class)를 사용하여 객체를 정의
- 객체지향언어마다 정의하는 문법이 다름
- Python은 class라는 예약어를 사용하고 클래스 명은 대문자로 시작

❖ 클래스를 이용하면 새로운 자료형들을 만들어 갈 수 있음
- 분수(fraction)를 다루어 보자.
  - ✓ 값(상태) : 분자, 분모
  - ✓ 행동: +, 비교(==), 출력(a/b 형태로 출력)

예) >>> my_fraction = Fraction(3,5)

```
>>> f1 = Fraction(1,4)
>>> f2 = Fraction(1,2)
>>> f3 = f1 + f2
>>> print(f3)
3/4
```

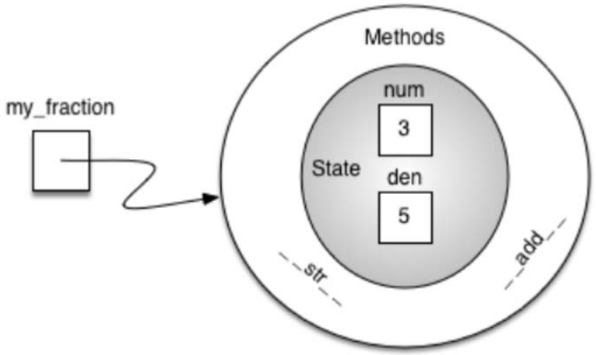


객체의 정의

❖ 클래스 Fraction (in Python)

참고: https://youtu.be/gFb9tvJZHXo

```
def gcd(m, n):
    while m % n != 0:
        m, n = n, m % n
    return n
```




```
class Fraction:
    """Class Fraction"""
    def __init__(self, top, bottom):
                """Constructor definition"""
        self.num = top
        self.den = bottom

    def __str__(self):
        return f"{self.num}/{self.den}"

    def __add__(self, other_fraction):
        new_num = self.num*other_fraction.den + self.den*other_fraction.num
        new_den = self.den*other_fractcion.den
        common = gcd(new_num, new_den)  ## 약분
        return Fraction(new_num//common, new_den//common)

    def __eq__(self, other_fraction):
        first_num = self.num * other_fraction.den
        second_num = other_fraction.num * self.den
        return first_num == second_num
```

초기화 및 상태값설정

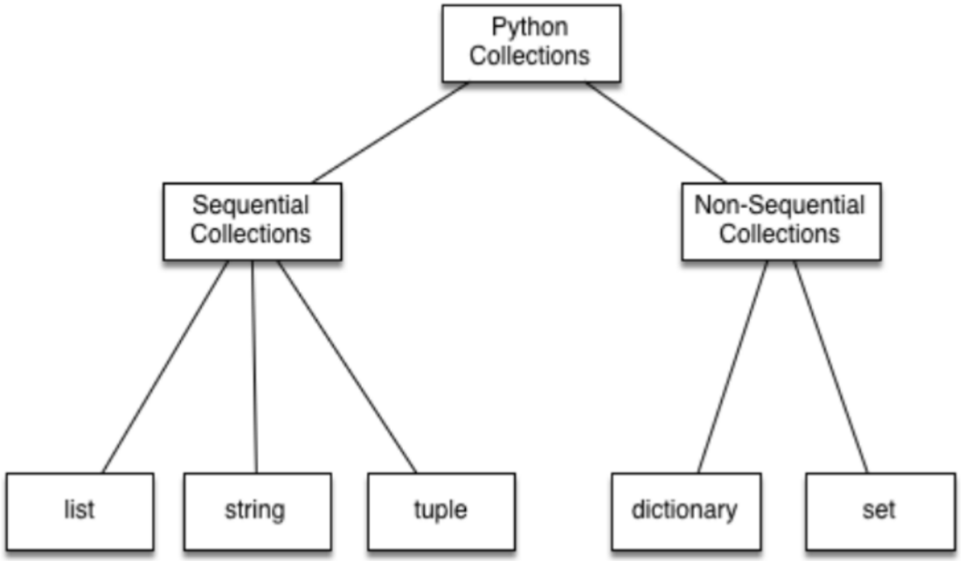
객체값을 출력하도록 만든 함수를 재정의 (a/b 형태출력)

더하기(+) 연산을 재정의

비교(==) 연산을 재정의

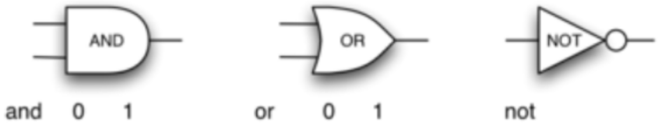
상속

- 객체를 사용하면 상위객체에서 설정된 내용을 하위 객체가 상속받아 사용할 수 있다.
- **상속(inheritance)**: 기본 클래스의 공통 기능을 물려받고, 다른 부분만 추가 또는 변경
  - 기본 클래스는 부모 클래스(또는 상위 클래스), Parent, Super, Base class 라고 부름
  - 기본 클래스 기능을 물려받는 클래스는 자식 클래스(또는 하위 클래스), Child, Sub, Derived class 라고 부름



Logic Gates의 연결

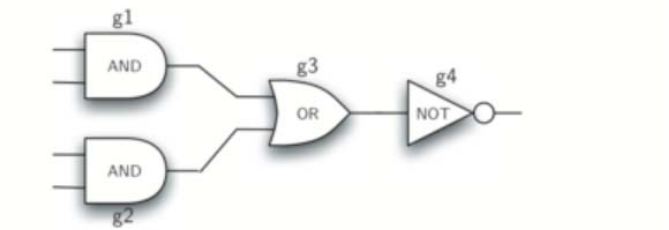
- ❖ 논리연산을 게이트들을 연결하여 논리 연산을 수행
- ❖ 논리 게이트를 클래스로 정의할 수 있음

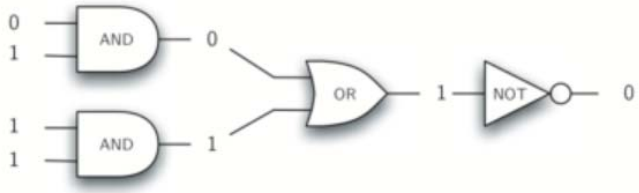


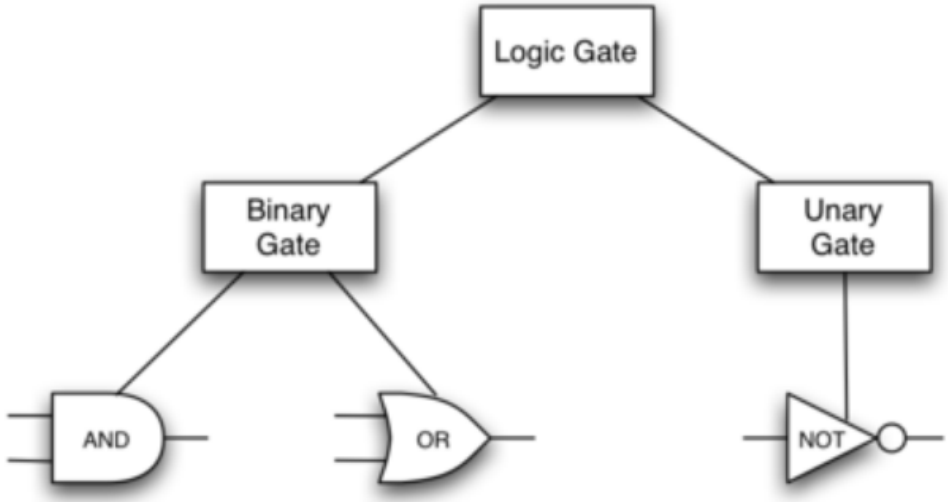
and	0	1
0	0	0
1	0	1

or	0	1
0	0	1
1	1	1

not	
0	1
1	0







KOREATECH 한국기술교육대학교

필요한 클래스 정의

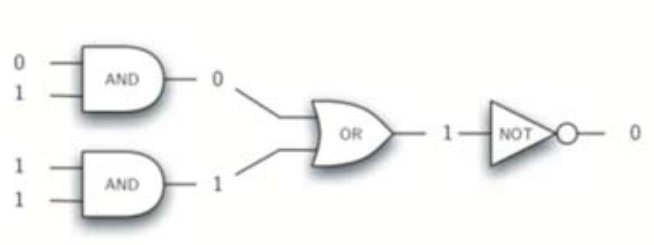
```
class LogicGate:

    def __init__(self, lbl):
        self.name = lbl
        self.output = None

    def get_label(self):
        return self.name

    def get_output(self):
        self.output = self.perform_gate_logic()
        return self.output
```

```
class BinaryGate(LogicGate):
    def __init__(self, lbl):
        super(BinaryGate, self).__init__(lbl)
        ## or LogicGate.__init__(lbl)
        self.pin_a = None
        self.pin_b = None
    def get_pin_a(self):
        if self.pin_a == None:
            return int(input("Enter pin A input for gate " + self.get_label() + ": "))
        else:
            return self.pin_a.get_from().get_output()
    def get_pin_b(self):
        if self.pin_b == None:
            return int(input("Enter pin B input for gate " + self.get_label() + ": "))
        else:
            return self.pin_b.get_from().get_output()
    def set_next_pin(self, source):
        if self.pin_a == None:
            self.pin_a = source
        else:
            if self.pin_b == None:
                self.pin_b = source
            else:
                print("Cannot Connect: NO EMPTY PINS on this gate")
```

필요한 클래스 정의

```
class AndGate(BinaryGate):

    def __init__(self, lbl):
        BinaryGate.__init__(self, lbl)

    def perform_gate_logic(self):
        a = self.get_pin_a()
        b = self.get_pin_b()
        if a == 1 and b == 1:
            return 1
        else:
            return 0
```

```
class OrGate(BinaryGate):

    def __init__(self, lbl):
        BinaryGate.__init__(self, lbl)

    def perform_gate_logic(self):
        a = self.get_pin_a()
        b = self.get_pin_b()
        if a == 1 or b == 1:
            return 1
        else:
            return 0
```




```
>>> g1 = AndGate("G1")
>>> g1.get_output()
Enter pin A input for gate G1: 1
Enter pin B input for gate G1: 0
0

>>> g2 = OrGate("G2")
>>> g2.get_output()
Enter pin A input for gate G2: 1
Enter pin B input for gate G2: 1
1
```

필요한 클래스 정의

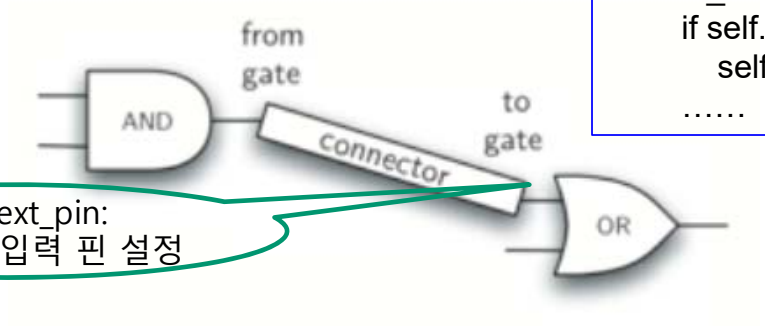
```
class UnaryGate(LogicGate):

    def __init__(self, lbl):
        LogicGate.__init__(self, lbl)

        self.pin = None

    def get_pin(self):
        if self.pin == None:
            return int(input("Enter pin input for gate " + self.get_label() + ": "))
        else:
            return self.pin.get_from().get_output()

    def set_next_pin(self, source):
        if self.pin == None:
            self.pin = source
        else:
            print("Cannot Connect: NO EMPTY PINS on this gate")
```

```
class NotGate(UnaryGate):

    def __init__(self, nlbl):
        UnaryGate.__init__(self, lbl)

    def perform_gate_logic(self):
        if self.get_pin():
            return 0
        else:
            return 1
```

필요한 클래스 정의

```
class Connector:

    def __init__(self, fgate, tgate):
        self.from_gate = fgate
        self.to_gate = tgate

        tgate.set_next_pin(self)

    def get_from(self):
        return self.from_gate

    def get_to(self):
        return self.to_gate
```

set_next_pin:
OR 게이트 입력 핀 설정



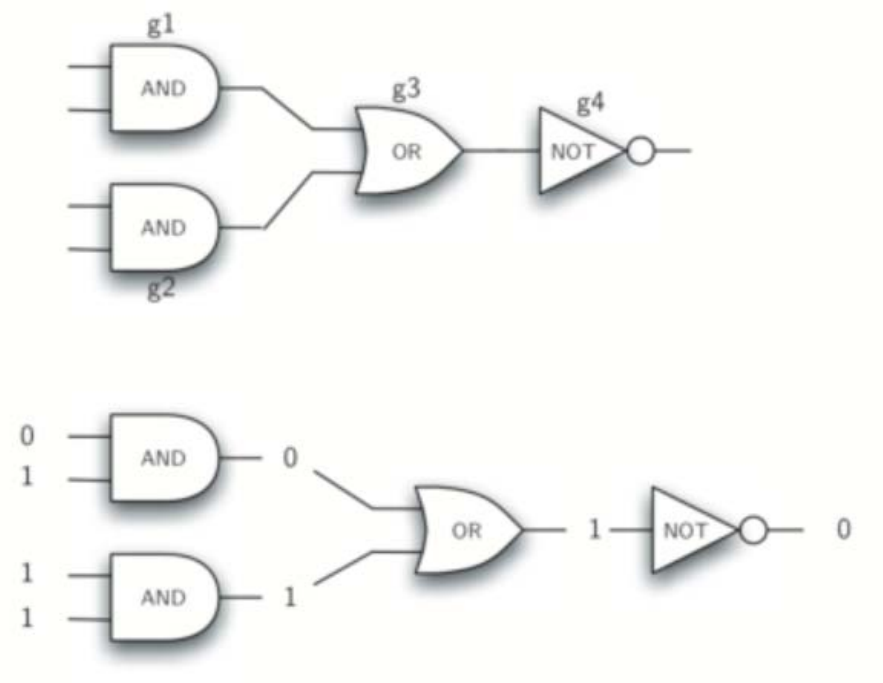

```
def set_next_pin(self, source):
        if self.pin_a == None:
            self.pin_a = source
        ……
```

```
def get_pin_a(self):
        if self.pin_a == None:
            return int(input("Enter pin A input for gate " + self.get_label() + ": "))
        else:
            return self.pin_a.get_from().get_output()
```

게이트의 입력을 직접 받거나 커넥터로 부터 받는다.
self.pin_a : 커넥터
self.pin_a.get_from : from gate
self.pin_a.get_from().get_output() : from gate 출력

결국 from gate의 출력이 to gate 입력으로 설정

Circuit 구성




```
def main():
    g1 = AndGate("G1")
    g2 = AndGate("G2")
    g3 = OrGate("G3")
    g4 = NotGate("G4")
    c1 = Connector(g1, g3)
    c2 = Connector(g2, g3)
    c3 = Connector(g3, g4)
    print(g4.get_output())

main()
```

참고: https://youtu.be/brrpvAlzOyM

파일과 모듈

파일

❖ 파일 핸들 제공: 운영체제에 무관하게 파일을 이 핸들을 통해서 다룸

nameHandle = open(‘kids’, ‘w’)

w -> write, r -> read, a -> append

```
nameHandle = open('kids', 'w')
for i in range(2):
    name = raw_input('Enter name: ')
    nameHandle.write(name + '\n')
nameHandle.close()
```

```
nameHandle = open('kids', 'r')
for line in nameHandle:
    print line
nameHandle.close()
```

```
nameHandle = open('kids', 'w')
nameHandle.write('Michael\n')
nameHandle.write('Mark\n')
nameHandle.close()
nameHandle = open('kids', 'r')
for line in nameHandle:
    print line[:-1]
nameHandle.close()
```

It will print

```
Michael
Mark
```

```
nameHandle = open('kids', 'a')
nameHandle.write('David\n')
nameHandle.write('Andrea\n')
nameHandle.close()
nameHandle = open('kids', 'r')
for line in nameHandle:
    print line[:-1]
nameHandle.close()
```

it will print

```
Michael
Mark
David
Andrea
```

파일연산

open(fn, 'w') fn is a string representing a file name. Creates a file for writing and returns a file handle.

open(fn, 'r') fn is a string representing a file name. Opens an existing file for reading and returns a file handle.

open(fn, 'a') fn is a string representing a file name. Opens an existing file for appending and returns a file handle.

fh.read() returns a string containing the contents of the file associated with the file handle fh.

fh.readline() returns the next line in the file associated with the file handle fh.

fh.readlines() returns a list each element of which is one line of the file associated with the file handle fh.

fh.write(s) write the string s to the end of the file associated with the file handle fh.

fh.writeLines(S) S is a sequence of strings. Writes each element of S to the file associated with the file handle fh.

fh.close() closes the file associated with the file handle fh.

모듈

- 프로그램이 커지면 한 파일에 모든 내용을 담을 수 없다.
- 팀으로 프로그램을 작성할 경우 각자가 담당한 부분을 개별 파일로 관리해야 한다.
- 모듈: 파이썬 파일(.py)로 defintions과 statements로 구성
  - 단일 파일(M.py)로 구성
  - 모듈은 import M으로 현재 프로그램 영역을 불러들인다.
  - 모듈내의 객체를 접근하기 위해선 M.x 형태로 dot notation 사용

- 파일 circle.py

```
pi = 3.14159

def area(radius):
    return pi*(radius**2)

def circumference(radius):
    return 2*pi*radius

def sphereSurface(radius):
    return 4.0*area(radius)

def sphereVolume(radius):
    return (4.0/3.0)*pi*(radius**3)
```

모듈 활용

```
import circle
print circle.pi
print circle.area(3)
print circle.circumference(3)
print circle.sphereSurface(3)
```

모듈

❖ 현재 프로그램의 scope 안에 모듈에 선언된 모든 객체를 동등하게 묶을(binding)수 있다.

from M import *

```
from circle import *
print pi
print circle.pi

NameError: name 'circle' is not defined
```

❖ 모듈안의 모든 값을 동등한 scope 수준으로 읽어들여서 처음 print pi는 3.14159를 출력하지만 두번째 print는 오류를 읽으킨다.

❖ 모듈을 어떻게 사용할지는 프로그래머가 결정해야 한다.

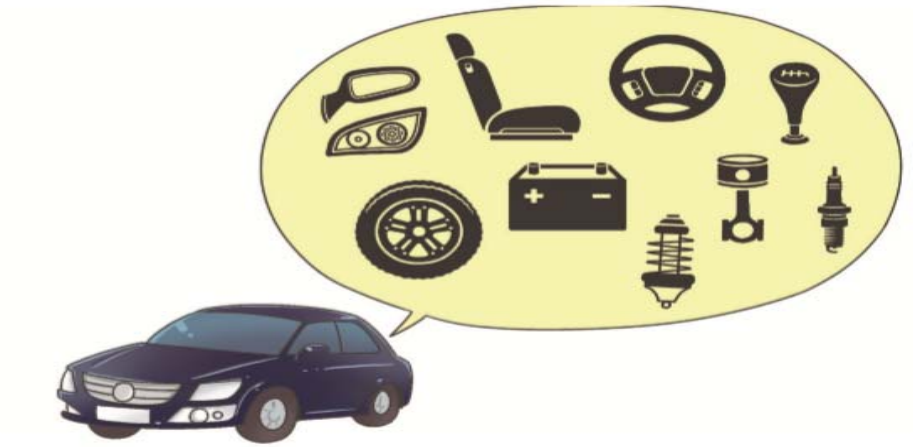
- 내부에 미리 선언된 변수나 함수들과 충돌 여부를 확신할 수 있을까?

❖ 모듈을 import하고 중간에 모듈의 코드를 수정하면?

- 처음 import시에 로드되어서 중간에 수정된 코드는 반영되지 않음
- reload()를 수행하며 수정사항을 반영할 수 있다.

모듈

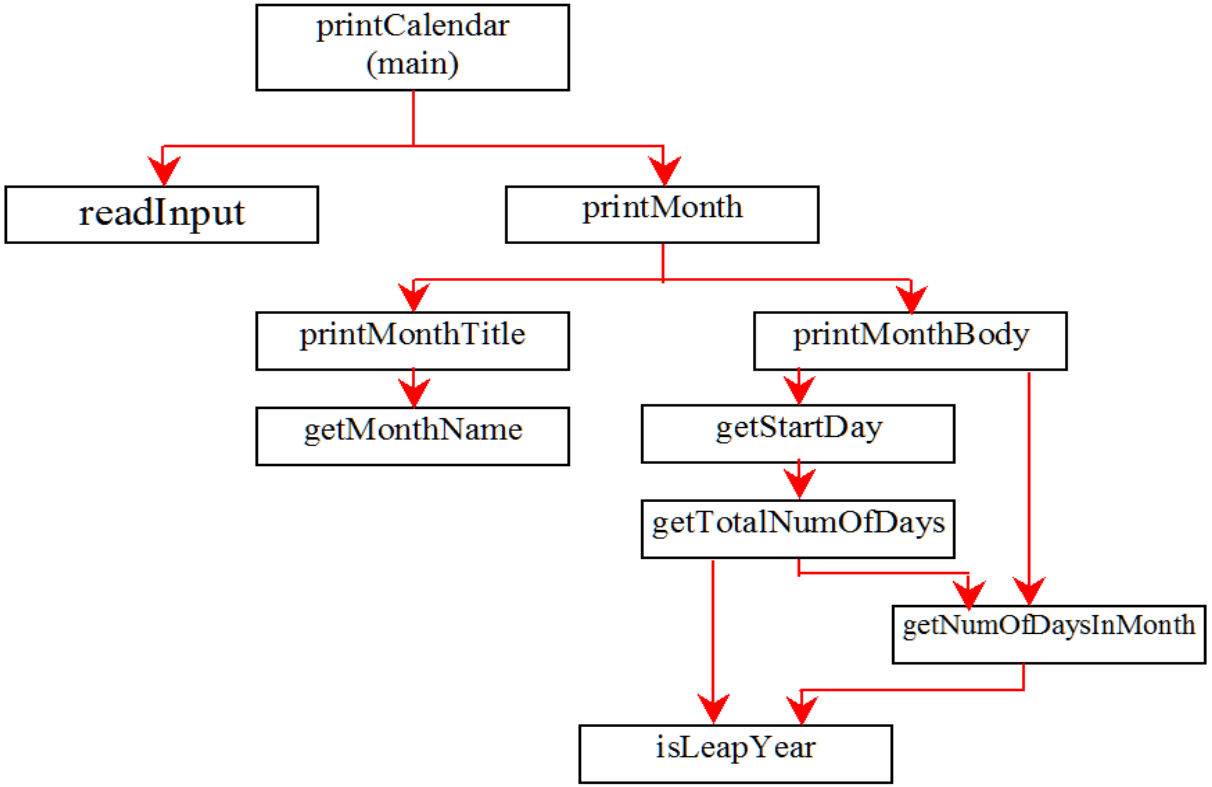
- 문제를 한 번에 해결하려고 하지 말고 더 작은 크기의 문제들로 분해한다. 문제가 충분히 작아질 때까지 계속해서 분해한다.
- 문제가 충분히 작아졌으면 각각의 문제를 모듈화한다
- 이들 모듈들을 조립하면 최종 프로그램이 완성된다.



Top-Down

❖ 적절히 pass keyword를 사용하여 top-down 개발에 사용

```
def printMonth(month)
    pass
( 지금은 pass 하고 나중에
 개발하겠음)
```



파이썬 디버거 pdb

- https://docs.python.org/2/library/pdb.html
- 인터프리터 내부에서 pdb 모듈을 불러서 실행

```
>>> import pdb
>>> import mymodule
>>> pdb.run('mymodule.test()')
> <string>(0)?()
(Pdb) continue
> <string>(1)?()
(Pdb) continue
NameError: 'spam'
> <string>(1)?()
(Pdb)
```

- python –m pdb myscript.py

c(ont(inue)	멈춰있는 program을 다시 run 하게 한다.
b(reak)	break point 를 설정한다.
n(ext)	다음 line 으로 넘어간다.
s(tep)	함수안으로 들어간다.
p expression	expression을 평가해서 값을 보여준다.
pp expression	expression 평가값을 pretty-print 로 보여준다.
l(ist) [first[, last]]	일정 구간의 source code 를 보여준다.

예외처리

예외처리

❖ 프로그램의 오류

test = [1,2,3]

test[3] → 인덱스 범위를 벗어남 → IndexError 라는 예외발생

❖ 일반적인 코드는 exception을 다루지 않음

```
successFailureRatio = numSuccesses/float(numFailures)
print 'The success/failure ratio is', successFailureRatio
print 'Now here'
```

→ numFailures 가 0(Zero) 일 수 있다

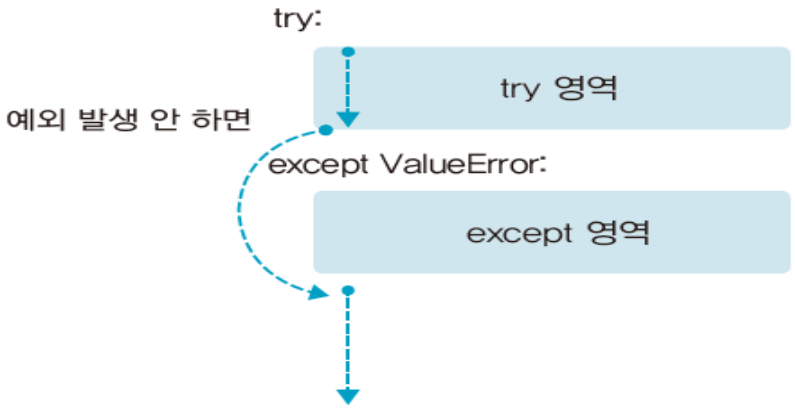
try block : exception을 발생할 수 있는 부분

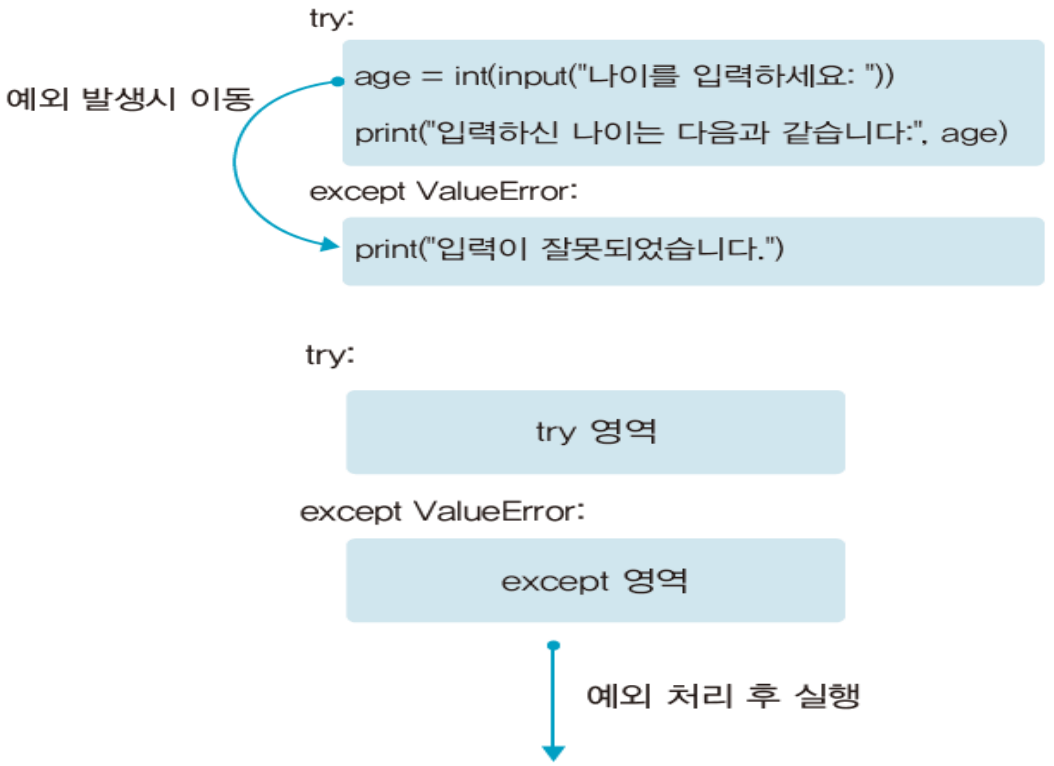
except block: exception이 발생할 경우 이를 처리하는 부분

```
try:
    successFailureRatio = numSuccesses/float(numFailures)
    print 'The success/failure ratio is', successFailureRatio
except ZeroDivisionError:
    print 'No failures so the success/failure ratio is undefined.'
print 'Now here'
```

예외처리

❖ 예외처리 실행





예외처리

❖ 정수의 square 값을 구하는 과정

```
while True:
    val = raw_input('Enter an integer: ')
    try:
        val = int(val)
        print 'The square of the number you entered is', val**2
        break #to exit the while loop
    except ValueError:
        print val, 'is not an integer'
```

❖ 코드를 함수를 사용해서 정리

```
def readInt():
    while True:
        val = raw_input('Enter an integer: ')
        try:
            val = int(val)
            return val
        except ValueError:
            print val, 'is not an integer'
```

val = readInt()

print ‘The square of the number you entered is’, val**2

강제 예외 발생

❖ 강제로 exception을 발생 시킬 수 있다.

방법) raise exceptionName(arguments)

여기서 exceptionName은
ValueError와 같은 내장된 예외를 사용할 수도 있고,
내장된 Exception class를 상속받아 프로그래머가
새롭게 정의한 예외일 수도 있다.

```
def getRatios(vect1, vect2):
    """Assumes: vect1 and vect2 are lists of equal length of numbers
       Returns: a list containing the meaningful values of
            vect1[i]/vect2[i]"""
    ratios = []
    for index in range(len(vect1)):
        try:
            ratios.append(vect1[index]/float(vect2[index]))
        except ZeroDivisionError:
            ratios.append(float('nan')) #nan = Not a Number
        except:
            raise ValueError('getRatios called with bad arguments')
    return ratios
```

강제 예외 처리

❖ 수행 결과

```
try:
    print getRatios([1.0,2.0,7.0,6.0], [1.0,2.0,0.0,3.0])
    print getRatios([], [])
    print getRatios([1.0, 2.0], [3.0])
except ValueError, msg:
    print msg
```

```
[1.0, 1.0, nan, 2.0]
[]
getRatios called with bad arguments
```

Assertions

- To Confirm that the state of the computation is as expected.
- 문법

  assert *Boolean_expression*

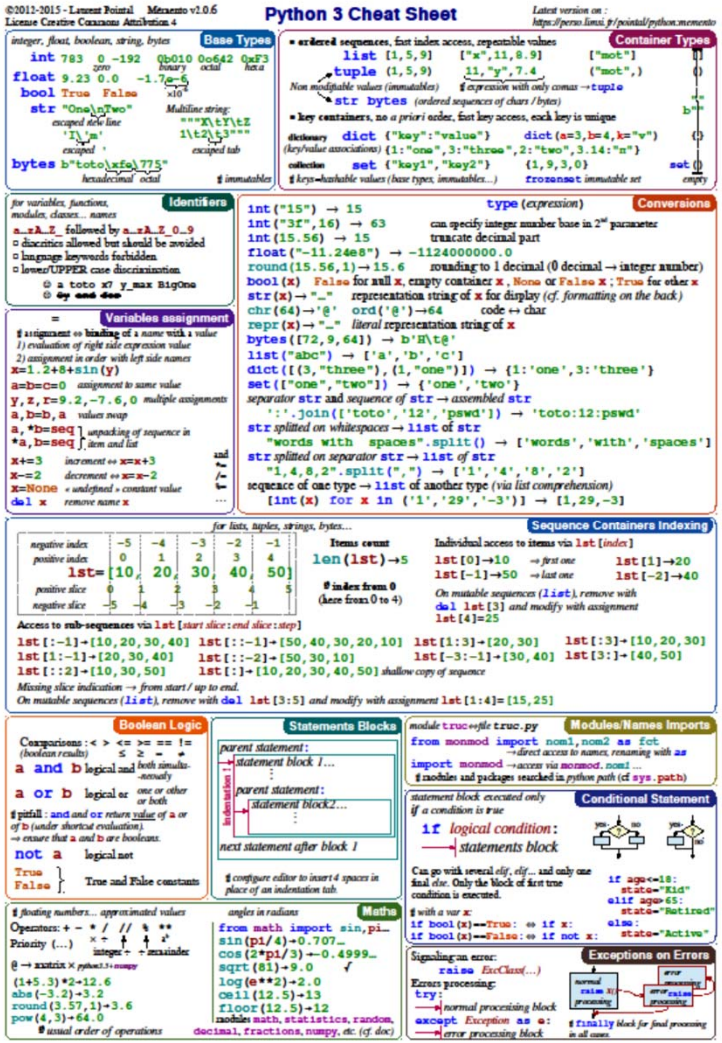
  assert *Boolean_expression*, argument

  Boolean_expression 을 평가해서 거짓일 경우 AssertionError 예외 발생

- Defensive programming tool

```
>>> number = input('Enter a positive number:')
Enter a positive number:-1
>>> assert (number > 0), 'Only positive numbers are allowed!'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AssertionError: Only positive numbers are allowed!
>>>
```

Cheat sheet 들이 많이 존재



KOREA TECH
한국기술교육대학교

Cheat sheet 들이 많이 존재

DATAQUEST

LEARN DATA SCIENCE ONLINE
Start Learning For Free - www.dataquest.io

Data Science Cheat Sheet

Pandas

KEY
We'll use shorthand in this cheat sheet
`df` - A pandas DataFrame object
`s` - A pandas Series object

IMPORTS
Import these to start
`import pandas as pd`
`import numpy as np`

IMPORTING DATA
`pd.read_csv(filename)` - From a CSV file
`pd.read_table(filename)` - From a delimited text file (like TSV)
`pd.read_excel(filename)` - From an Excel file
`pd.read_sql(query, connection_object)` - Reads from a SQL table/database
`pd.read_json(json_string)` - Reads from a JSON formatted string, URL or file.
`pd.read_html(url)` - Parses an html URL, string or file and extracts tables to a list of dataframes
`pd.read_clipboard()` - Takes the contents of your clipboard and passes it to `read_table()`
`pd.DataFrame(dict)` - From a dict, keys for columns names, values for data as lists

EXPORTING DATA
`df.to_csv(filename)` - Writes to a CSV file
`df.to_excel(filename)` - Writes to an Excel file
`df.to_sql(table_name, connection_object)` - Writes to a SQL table
`df.to_json(filename)` - Writes to a file in JSON format
`df.to_html(filename)` - Saves as an HTML table
`df.to_clipboard()` - Writes to the clipboard

CREATE TEST OBJECTS
Useful for testing
`pd.DataFrame(np.random.rand(20,5))` - 5 columns and 20 rows of random floats
`pd.Series(my_list)` - Creates a series from an iterable `my_list`
`df.index = pd.date_range('1900/1/30', periods=df.shape[0])` - Adds a date index

VIEWING/INSPECTING DATA
`df.head(n)` - First n rows of the DataFrame
`df.tail(n)` - Last n rows of the DataFrame
`df.shape()` - Number of rows and columns
`df.info()` - Index, Datatype and Memory information
`df.describe()` - Summary statistics for numerical columns
`s.value_counts(dropna=False)` - Views unique values and counts
`df.apply(pd.Series.value_counts)` - Unique values and counts for all columns

SELECTION
`df[col]` - Returns column with label `col` as Series
`df[[col1, col2]]` - Returns Columns as a new DataFrame
`s.iloc[0]` - Selection by position
`s.loc[0]` - Selection by index
`df.iloc[0,:]` - First row
`df.iloc[0,0]` - First element of first column

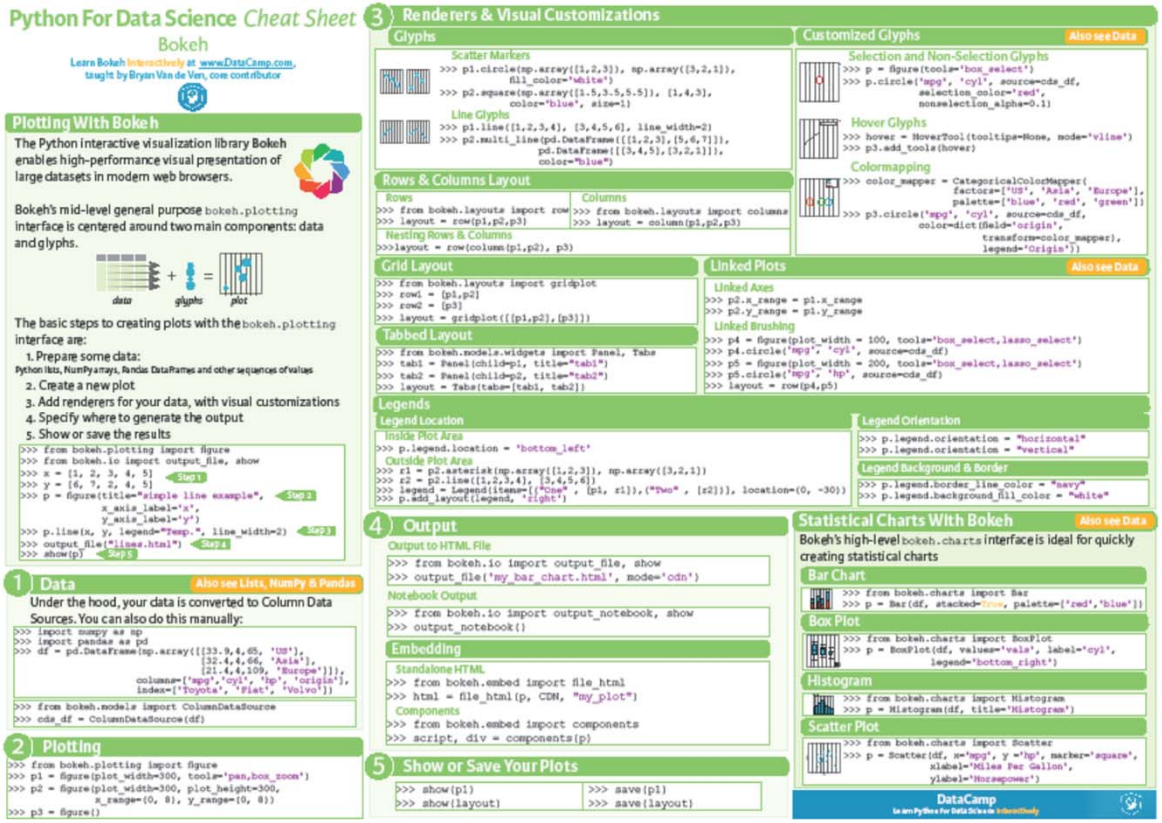
DATA CLEANING
`df.columns = ['a','b','c']` - Renames columns
`pd.isnull()` - Checks for null Values, Returns Boolean Array
`pd.notnull()` - Opposite of `s.isnull()`
`df.dropna()` - Drops all rows that contain null values
`df.dropna(axis=1)` - Drops all columns that contain null values
`df.dropna(axis=1,thresh=n)` - Drops all rows have have less than n non null values
`df.fillna(x)` - Replaces all null values with x
`s.fillna(s.mean())` - Replaces all null values with the mean (mean can be replaced with almost any function from the statistics section)
`s.astype(float)` - Converts the datatype of the series to float
`s.replace(1,'one')` - Replaces all values equal to 1 with 'one'
`s.replace([1,3],['one','three'])` - Replaces all 1 with 'one' and 3 with 'three'
`df.rename(columns=lambda x: x + 1)` - Mass renaming of columns
`df.rename(columns={'old_name': 'new_name'})` - Selective renaming
`df.set_index('column_one')` - Changes the index
`df.rename(index=lambda x: x + 1)` - Mass renaming of index

FILTER, SORT, & GROUPBY
`df[df[col] > 0.5]` - Rows where the `col` column is greater than `0.5`
`df[(df[col] > 0.5) & (df[col] < 0.7)]` - Rows where `0.7 > col > 0.5`
`df.sort_values(col1)` - Sorts values by `col1` in ascending order
`df.sort_values(col2,ascending=False)` - Sorts values by `col2` in descending order
`df.sort_values([col1,col2], ascending=[True,False])` - Sorts values by `col1` in ascending order then `col2` in descending order
`df.groupby(col)` - Returns a groupby object for values from one column
`df.groupby([col1,col2])` - Returns a groupby object values from multiple columns
`df.groupby(col1)[col2].mean()` - Returns the mean of the values in `col2`, grouped by the values in `col1` (mean can be replaced with almost any function from the statistics section)
`df.pivot_table(index=col1,values=[col2,col3],aggfunc=mean)` - Creates a pivot table that groups by `col1` and calculates the mean of `col2` and `col3`
`df.groupby(col1).agg(np.mean)` - Finds the average across all columns for every unique column 1 group
`df.apply(np.mean)` - Applies a function across each column
`df.apply(np.max, axis=1)` - Applies a function across each row

JOIN/COMBINE
`df1.append(df2)` - Adds the rows in `df1` to the end of `df2` (columns should be identical)
`pd.concat([df1, df2],axis=1)` - Adds the columns in `df1` to the end of `df2` (rows should be identical)
`df1.join(df2,on=col1,how='inner')` - SQL-style joins the columns in `df1` with the columns on `df2` where the rows for `col` have identical values. `how` can be one of `'left'`, `'right'`, `'outer'`, `'inner'`

STATISTICS
These can all be applied to a series as well.
`df.describe()` - Summary statistics for numerical columns
`df.mean()` - Returns the mean of all columns
`df.corr()` - Returns the correlation between columns in a DataFrame
`df.count()` - Returns the number of non-null values in each DataFrame column
`df.max()` - Returns the highest value in each column
`df.min()` - Returns the lowest value in each column
`df.median()` - Returns the median of each column
`df.std()` - Returns the standard deviation of each column

LEARN DATA SCIENCE ONLINE
Start Learning For Free - www.dataquest.io



Python For Data Science *Cheat Sheet*

Bokeh

Learn Bokeh interactively at www.DataCamp.com, taught by Bryan Van de Ven, core contributor

Plotting With Bokeh

The Python interactive visualization library Bokeh enables high-performance visual presentation of large datasets in modern web browsers.

Bokeh's mid-level general purpose `bokeh.plotting` interface is centered around two main components: data and glyphs.

data + glyphs = plot

The basic steps to creating plots with the `bokeh.plotting` interface are:

1. Prepare some data:
Python lists, NumPy arrays, Pandas DataFrames and other sequences of values
2. Create a new plot
3. Add renderers for your data, with visual customizations
4. Specify where to generate the output
5. Show or save the results

```
>>> from bokeh.plotting import figure
>>> from bokeh.io import output_file, show
>>> x = [1, 2, 3, 4, 5]                                  Step 1
>>> y = [6, 7, 2, 4, 5]
>>> p = figure(title="simple line example",               Step 3
              x_axis_label='x',
              y_axis_label='y')
>>> p.line(x, y, legend="Temp.", line_width=2)           Step 3
>>> output_file("lines.html")                            Step 4
>>> show(p)                                              Step 5
```

1 Data

Also see Lists, NumPy & Pandas

Under the hood, your data is converted to Column Data Sources. You can also do this manually:

```
>>> import numpy as np
>>> import pandas as pd
>>> df = pd.DataFrame(np.array([[33.9,4,65, 'US'],
                                [32.4,4,66, 'Asia'],
                                [21.4,4,109, 'Europe']]),
                      columns=['mpg','cyl', 'hp', 'origin'],
                      index=['Toyota', 'Fiat', 'Volvo'])
>>> from bokeh.models import ColumnDataSource
>>> cds_df = ColumnDataSource(df)
```

2 Plotting

```
>>> from bokeh.plotting import figure
>>> p1 = figure(plot_width=300, tools='pan,box_zoom')
>>> p2 = figure(plot_width=300, plot_height=300,
                x_range=(0, 8), y_range=(0, 8))
>>> p3 = figure()
```

3 Renderers & Visual Customizations

Glyphs

Scatter Markers
```
>>> p1.circle(np.array([1,2,3]), np.array([3,2,1]),
              fill_color='white')
>>> p2.square(np.array([1.5,3.5,5.5]), [1,4,3],
              color="blue", size=1)
```

Line Glyphs
```
>>> p1.line([1,2,3,4], [3,4,5,6], line_width=2)
>>> p2.multi_line(pd.DataFrame([[1,2,3],[5,6,7]]),
                  pd.DataFrame([[3,4,5],[3,2,1]]),
                  color="blue")
```

Customized Glyphs — Also see Data

Selection and Non-Selection Glyphs
```
>>> p = figure(tools='box_select')
>>> p.circle('mpg', 'cyl', source=cds_df,
             selection_color='red',
             nonselection_alpha=0.1)
```

Hover Glyphs
```
>>> hover = HoverTool(tooltips=None, mode='vline')
>>> p3.add_tools(hover)
```

Colormapping
```
>>> color_mapper = CategoricalColorMapper(
                   factors=['US', 'Asia', 'Europe'],
                   palette=['blue', 'red', 'green'])
>>> p3.circle('mpg', 'cyl', source=cds_df,
              color=dict(field='origin',
                         transform=color_mapper),
              legend='Origin'))
```

Rows & Columns Layout

Rows
```
>>> from bokeh.layouts import row
>>> layout = row(p1,p2,p3)
```

Columns
```
>>> from bokeh.layouts import column
>>> layout = column(p1,p2,p3)
```

Nesting Rows & Columns
```
>>>layout = row(column(p1,p2), p3)
```

Grid Layout
```
>>> from bokeh.layouts import gridplot
>>> row1 = [p1,p2]
>>> row2 = [p3]
>>> layout = gridplot([[p1,p2],[p3]])
```

Tabbed Layout
```
>>> from bokeh.models.widgets import Panel, Tabs
>>> tab1 = Panel(child=p1, title="tab1")
>>> tab2 = Panel(child=p2, title="tab2")
>>> layout = Tabs(tabs=[tab1, tab2])
```

Linked Plots — Also see Data

Linked Axes
```
>>> p2.x_range = p1.x_range
>>> p2.y_range = p1.y_range
```

Linked Brushing
```
>>> p4 = figure(plot_width = 100, tools='box_select,lasso_select')
>>> p4.circle('mpg', 'cyl', source=cds_df)
>>> p5 = figure(plot_width = 200, tools='box_select,lasso_select')
>>> p5.circle('mpg', 'hp', source=cds_df)
>>> layout = row(p4,p5)
```

Legends

Legend Location

Inside Plot Area
```
>>> p.legend.location = 'bottom_left'
```

Outside Plot Area
```
>>> r1 = p2.asterisk(np.array([1,2,3]), np.array([3,2,1])
>>> r2 = p2.line([1,2,3,4], [3,4,5,6])
>>> legend = Legend(items=[("One" , [p1, r1]),("Two" , [r2])], location=(0, -30))
>>> p.add_layout(legend, 'right')
```

Legend Orientation
```
>>> p.legend.orientation = "horizontal"
>>> p.legend.orientation = "vertical"
```

Legend Background & Border
```
>>> p.legend.border_line_color = "navy"
>>> p.legend.background_fill_color = "white"
```

4 Output

Output to HTML File
```
>>> from bokeh.io import output_file, show
>>> output_file('my_bar_chart.html', mode='cdn')
```

Notebook Output
```
>>> from bokeh.io import output_notebook, show
>>> output_notebook()
```

Embedding

Standalone HTML
```
>>> from bokeh.embed import file_html
>>> html = file_html(p, CDN, "my_plot")
```

Components
```
>>> from bokeh.embed import components
>>> script, div = components(p)
```

5 Show or Save Your Plots

```
>>> show(p1)
>>> show(layout)
```

```
>>> save(p1)
>>> save(layout)
```

Statistical Charts With Bokeh

Also see Data

Bokeh's high-level `bokeh.charts` interface is ideal for quickly creating statistical charts

Bar Chart
```
>>> from bokeh.charts import Bar
>>> p = Bar(df, stacked=True, palette=['red','blue'])
```

Box Plot
```
>>> from bokeh.charts import BoxPlot
>>> p = BoxPlot(df, values='vals', label='cyl',
                legend='bottom_right')
```

Histogram
```
>>> from bokeh.charts import Histogram
>>> p = Histogram(df, title="Histogram")
```

Scatter Plot
```
>>> from bokeh.charts import Scatter
>>> p = Scatter(df, x='mpg', y ='hp', marker='square',
                xlabel='Miles Per Gallon',
                ylabel='Horsepower')
```

DataCamp
Learn Python For Data Science Interactively

파이썬 예제

Covid-19 관련 예제

❖ Using Python to visualize COVID-19 projections (https://opensource.com/article/20/4/python-data-covid-19)

- download data

```
import pycountry
import plotly.express as px
import matplotlib.pyplot as plt
import pandas as pd
URL_DATASET = r'https://raw.githubusercontent.com/datasets/covid-19/master/data/countries-aggregated.csv'
df1 = pd.read_csv(URL_DATASET)
print(df1.head(3))  # Get first 3 entries in the dataframe
print(df1.tail(3))  # Get last 3 entries in the dataframe
```

- Select data for India → 한국은?

```
#### ----- Step 2 (Select data for India)----
df_india = df1[df1['Country'] == 'India']
print(df_india.head(3))
```

Covid-19 관련 예제

❖ Using Python to visualize COVID-19 projections (https://opensource.com/article/20/4/python-data-covid-19)

- Plot data

```
#### ----- Step 3 (Plot data)----
# Increase size of plot
plt.rcParams["figure.figsize"]=20,20  # Remove if not on Jupyter
# Plot column 'Confirmed'
df_india.plot(kind = 'bar', x = 'Date', y = 'Confirmed', color = 'blue')

ax1 = plt.gca()
df_india.plot(kind = 'bar', x = 'Date', y = 'Deaths', color = 'red', ax = ax1)
plt.show()
```

❖ 국가를 확대해서 결과를 애니메이션으로 보여주기

학습정리

- 함수
- 객체
- 파일과 모듈
- 예외처리