함수 - function

- 함수는 하나의 관련 작업을 수행하는 데 사용되는 재사용 가능한 코드 블록이다.
  - 함수는 응용 프로그램의 모듈성을 높이고 코드 재사용 수준을 높인다.
- Python은 기본적으로 많은 내장 함수를 제공한다.
- 문자열 등을 다루면서 함수를 사용한 경험을 상기해 보자.
  - len("Python programming")
- 필요에 따라 함수를 생성하여 원하는 기능을 따로 분리하여 사용하는 법을 익혀보자.
  - Decomposition and abstraction
- Python에서 제공하는 함수가 아니라 사용자 정의의 새로운 함수를 만들 수 있다.

함수 정의의 규칙

- 함수 블록은 키워드 def와 함수 이름 및 괄호()로 시작한다.
- 모든 입력 매개 변수는 괄호 안에 있어야 한다. 이 괄호 안에 매개 변수를 정의할 수도 있다.
- 모든 함수 내의 코드 블록은 콜론(:)으로 시작되고 들여쓰기를 하여야 한다.
- return [expression] 명령문은 함수를 종료하고 호출자에게 표현식을 전달한다.

def function_name(< argument >):
    < something to do >

함수 예제

- 제곱 후 1을 더해서 반환하는 함수

In [1]:

```
def square_plus_one(x):
    ans = x**2 + 1
    return ans
```

- x라는 인자(argument)를 받아서
- x**2 + 1 의 계산을 수행한 후
- 결과를 ans에 대입하여 반환(return)하고 있음
- 콘솔이나 에디터에서 square_plus_one(10) 등을 수행해 보자.

In [2]:

```
square_plus_one(10)
```

Out[2]:

101

함수 예제(2)

- 제곱근 반환 함수

In [3]:

```
def my_sqrt(x):
    ans = 0
    increment = 1
    while increment >= 0.0001:
          ans += increment
          if ans**2 == x:
                break
          elif ans**2>x:
                  ans -= increment
                  increment = increment/10.0
    return ans
```

함수 사용

- 함수를 정의하였으면 이를 스크립트 내에서 활용할 수 있음

In [6]:

```
print(my_sqrt(2))
```

1.4142

In [7]:

```
y = 16
print(my_sqrt(y))
```

4

In [8]:

```
z = 12
print(my_sqrt(z*12))
```

12

In [9]:

```
x = my_sqrt(100)
print(x*x)
```

100

함수의 인자(argument)

- 함수는 아무런 인자를 받지 않을 수 있음

In [8]:

```
def print_something():
    print("This function does nothing.")
```

- 위 함수는 아무런 인자를 받지 않을 뿐만 아니라, 아무런 반환값도 없음
  - print_something() 함수를 실행해 보자.

In [9]:

```
print_something()
```

```
This function does nothing.
```

- 인자가 필요하지만 함수 호출 시 입력하지 않았을 경우 에러가 발생한다.

In [12]:

```
def printme( str ):
    print("Input is :",str)
    return;

printme()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-12-909177142d6d> in <module>()
      3     return;
      4
----> 5 printme()

TypeError: printme() missing 1 required positional argument: 'str'
```

- 다음과 같이 인자가 어떤 값을 입력받는지 확실히 표현할 수도 있다.

In [14]:

```
def printme( str ):
    print("Input is :",str)
    return;


printme( str = "My string")
```

```
Input is : My string
```

여러 개의 인자를 받는 함수

- 함수는 여러 개의 인자를 받을 수 있다.

In [16]:

```
def printinfo( name, age ):
    print("Name: ", name)
    print("Age: ", age)
    return;

printinfo( age=50, name="miki" )
```

```
Name:  miki
Age:  50
```

In [13]:

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

In [14]:

```
my_max(10,12)
```

Out[14]:

12

여러 개의 인자를 받는 함수(2)

- 최대공약수를 구하는 함수

In [25]:

```
def my_GCD(x, y):
    j = 1
    ans = 1
    while j <= x and j <= y:
        if x%j == 0 and y%j == 0:
            ans = j
        j += 1
    return ans
```

In [26]:

```
my_GCD(24, 84)
```

Out[26]:

12

예제

In [18]:

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

- 다음은 동치

In [19]:

```
printName('Kyungsub', 'Lee', False)
printName('Kyungsub', 'Lee', reverse=False)
printName(lastName='Lee', firstName='Kyungsub', reverse=False)
```

```
Kyungsub Lee
Kyungsub Lee
Kyungsub Lee
```

default parameter

- 함수 생성 시 다음과 같이 default parameter를 설정할 수 있다.
- default parameter는 입력값이 따로 없을 때, 미리 설정된 default값을 이용한다.

In [2]:

```
def printName(firstName, lastName, reverse = False):
    if reverse:
        print(lastName + ', ' + firstName)
    else:
        print(firstName, lastName)

printName('Kyungsub', 'Lee')
```

```
Kyungsub Lee
```

In [3]:

```
printName('Kyungsub', 'Lee', True)
```

```
Lee, Kyungsub
```

In [28]:

```
def printinfo( name, age = 35 ):
    print("Name: ", name)
    print("Age: ", age)
    return
```

In [29]:

```
printinfo( age=50, name="miki" )
printinfo( name="miki" )
```

```
Name:  miki
Age:  50
Name:  miki
Age:  35
```

지역 변수(local variable)

- 함수 내의 변수는 함수 내에서만 존재하는 변수

In [22]:

```
def my_func():
    x = 20

x = 10
my_func()
print(x)
```

10

- 함수 내의 지역 변수와 함수 바깥의 전역 변수(global variable)이 분리되어 있음을 주의

In [23]:

```
def increase(x):
    x = x + 1

x=1
increase(x)
print(x)
```

1

In [24]:

```
def my_func():
    x = 20

x = 10
my_func()
print(x)
```

10

In [25]:

```
def increase(x):
    x = x + 1

x = 1
increase(x)
print(x)
```

1

In [26]:

```
def my_func():
    x = 20
    return x

x = 10
x = my_func()
print(x)
```

20

In [27]:

```
def increase(x):
    x = x + 1
    return x

x = 1
x = increase(x)
print(x)
```

2

농장 문제

- 농장에 돼지들과 닭들이 있다. 총 머리의 수는 20개, 총 다리의 수는 56개이다. 돼지와 닭의 수는 각각 얼마인가?
  - Brute force 방법으로 풀어보자.

In [28]:

```
for x in range(0,21):
    y = 20 - x
    if 4*x + 2*y == 56:
        break
print(x, y)
```

8 12

농장 문제 – 함수

- 임의의 머리 수와 다리 수가 주어졌을 때, 돼지와 닭의 수를 구하는 함수를 작성해 보자.

In [29]:

```
def farm(numHeads, numLegs):
    for x in range(0, numHeads + 1):
        y = numHeads - x
        if 4*x + 2*y == numLegs:
            return [x, y]
```

In [31]:

```
print(farm(20, 56))
```

[8, 12]

In [32]:

```
print(farm(20, 60))
```

[10, 10]

None

- Python에서 없는 값을 나타내기 위해 사용되는 값
- None을 이용하여 해가 없는 경우를 대비하여 보자.

In [33]:

```
def farm(numHeads, numLegs):
    for x in range(0, numHeads + 1):
         y = numHeads - x
         if 4*x + 2*y == numLegs:
              return [x, y]
    return [None, None]

farm(100,10)
```

Out[33]:

```
[None, None]
```

- 앞의 예제에서 return이 두 번 등장함을 볼 수 있음

In [34]:

```
def farm(numHeads, numLegs):
    for x in range(0, numHeads + 1):
        y = numHeads - x
        if 4*x + 2*y == numLegs:
             return [x, y]
    return [None, None]
```

- 첫 번째 return은 방정식의 조건이 만족하면 다리의 숫자를 return하고 함수를 종료
- 두 번째 return은 방정식의 조건을 만족하는 해가 없을 때 None을 반환

함수를 호출하는 함수

In [37]:

```
def farm_UI():
    heads = int(input('Enter the number of heads : '))
    legs = int(input('Enther the number of legs : '))
    [pigs, chickens] = farm(heads, legs)
    if pigs == None:
          print('No solution')
    else:
          print('Number of pigs : ', pigs)
          print('Number of chickens : ', chickens)

farm_UI()
```

```
Enter the number of heads : 12
Enther the number of legs : 24
Number of pigs :  0
Number of chickens :  12
```

Built-in function

- Python에 이미 내장되어 있는 built-in function들을 활용할 수 있음
- https://docs.python.org/2/library/functions.html (https://docs.python.org/2/library/functions.html)
- range(), str(), raw_input(), len(), …
- max(), min(), abs(), type(), …

Module

- 기본적으로 제공하는 Built-in function 외에도 module을 통하여 다양한 함수 제공
- 예
  - math 모듈
    - 기본적인 수학 연산을 도와주는 모듈
    - https://docs.python.org/2/library/math.html (https://docs.python.org/2/library/math.html)
  - random 모듈
    - 다양한 분포에 대한 random number를 활용할 수 있게 해주는 모듈
    - https://docs.python.org/2/library/random.html (https://docs.python.org/2/library/random.html)
- import 명령을 사용하여 모듈을 불러와 사용

Math module

- import math
- math.sqrt(x)
  - x의 제곱근 반환
- math.exp(x)
  - e**x 반환
- math.sin(x)
  - sin 함수값 반환
- math.pi
  - 3.141592…

In [38]:

```
import math
math.sqrt(71)
```

Out[38]:

8.426149773176359

In [39]:

```
math.sin(math.pi/2)
```

Out[39]:

1.0

division - 나누기

- 다음은 Python 3.x에서는 해당되지 않는 내용이다.
- Python 2.x에서는 나누기(/)를 사용시 기본적으로 정수를 먼저 반환하려고 함.
  - 5/2 = 2
- 실수 나누기를 하고 싶은 경우
  - 5/2.0 혹은 5/float(2)
- 만약 default로 실수를 반환하는 나누기(/)를 사용하고 싶을 때, 다음 import문을 사용할 수 있음

In [40]:

```
from __future__ import division
5/2
```

Out[40]:

2.5

In [41]:

```
5//2
```

Out[41]:

2

Random module

- import random
- random.random()
  - 0이상 1이하의 랜덤값 반환
- random.uniform(a,b)
  - a이상 b이하에서 uniform 분포를 가지는 랜덤값 반환
- random.randint(a,b)
  - a이상 b이하의 랜덤 정수 반환
- random.gauss(mu, sigma), random.normalvariate(mu, sigma)
  - 평균 mu, 표준편차 sigma의 정규 분포를 가지는 랜덤값 반환

In [42]:

```
import random
random.random()
```

Out[42]:

0.6762294131998526

In [43]:

```
random.uniform(0, 10)
```

Out[43]:

1.8447516467769853

In [44]:

```
random.randint(1, 5)
```

Out[44]:

4

In [45]:

```
random.gauss(0, 1)
```

Out[45]:

-0.7165523306675587

t – 랜덤 변수

- gauss() 함수와 gammavariate() 함수를 이용하여 t-분포를 따르는 랜덤 변수를 생성하는 함수

In [46]:

```
import math
import random

def student_t(nu): # nu equals number of degrees of freedom
    x = random.gauss(0.0, 1.0)
    y = 2.0*random.gammavariate(0.5*nu, 2.0)
    return x / (math.sqrt(y/nu))
```

In [47]:

```
student_t(10)
```

Out[47]:

-0.20820980729502514

모듈에 대한 여러 가지

- 파이썬 스크립트 파일로 모듈을 만들 수 있음
  - 기본 작업 directory에 저장하여 import 할 수 있음
- 모듈 변경과 갱신
  - 혹시 모듈을 변경하였다면 reload 명령을 이용하여 갱신
    - reload(fibo)
- 모듈 파일을 불러오는 다양한 방법

In [52]:

```
import random
random.randint(1, 10)
```

Out[52]:

7

In [53]:

```
from random import *
randint(1, 10)
```

Out[53]:

6

In [54]:

```
import random as r
r.randint(1, 10)
```

Out[54]:

3