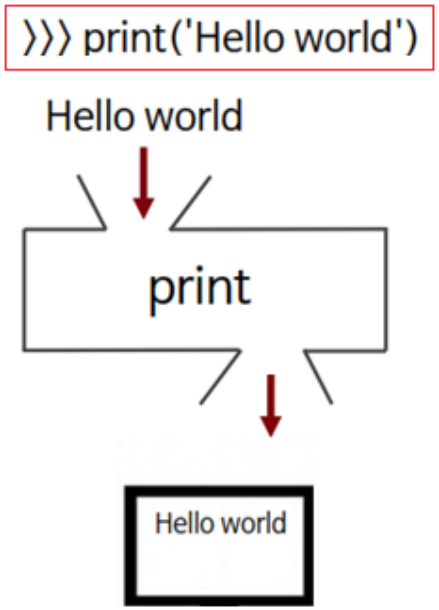
창의적 사고와 코딩

Lecture 5. 함수

함수의 기본 구조

함수(Function)란?

- 함수는 특정 작업을 수행하는 명령어들의 모음에 이름을 붙인 것
- 함수 안의 명령어들을 실행하려면 함수를 호출(call)하면 된다.
- black box
  - 주어진 입력에 대해서 어떤 과정을 거쳐 출력이 나오는지가 숨겨져 있다.
  - 함수는 작업에 필요한 데이터를 전달받을 수 있으며, 작업이 완료된 후에는 작업의 결과를 호출자에게 반환할 수 있다.
- 함수 이름, 입력, 출력이 중요



함수의 필요성


```
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
```

파이썬 함수

- **내장 함수(built-in functions)**
  - 파이썬 언어에서 미리 만들어서 제공하는 함수들
  - IDLE에서 dir(__builtings__)라고 입력하면 파이썬에서 제공하는 내장 함수 목록을 볼 수 있다.
  - 내장 함수에 어떤 것들이 있는지 학습하고 적절히 사용할 줄 아는 것이 중요
- **사용자 정의 함수(user-defined functions)**
  - 사용자가 직접 만드는 함수
  - 함수 작성 문법을 익히고 직접 작성해 보는 것이 중요

내장 함수(Built-in Function)

- Python에서 제공되는 내장 함수(Built-in Function)는 별도의 모듈(Model)의 추가(import 문) 없이 사용할 수 있는 함수들로 대략 76가지의 함수들이 있다.
- 그 중 가장 대표적인 함수로는 입출력 함수인 print(), input()이 있으며 그 외에도 len(), range(), int()등 다양한 함수들이 있다.
- Python에서 제공되는 내장함수들은 다음 주소의 홈페이지에서 확인할 수 있다.
  https://docs.python.org/3.8/library/functions.html

내장 함수(Built-in Function)

```
>>> dir(__builtins__)
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BlockingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProcessError', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedError', 'ConnectionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EnvironmentError', 'Exception', 'False', 'FileExistsError', 'FileNotFoundError', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarning', 'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'ModuleNotFoundError', 'NameError', 'None', 'NotADirectoryError', 'NotImplemented', 'NotImplementedError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'PermissionError', 'ProcessLookupError', 'RecursionError', 'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning', 'StopAsyncIteration', 'StopIteration', 'SyntaxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarning', 'ValueError', 'Warning', 'WindowsError', 'ZeroDivisionError', '__build_class__', '__debug__', '__doc__', '__import__', '__loader__', '__name__', '__package__', '__spec__', 'abs', 'all', 'any', 'ascii', 'bin', 'bool', 'breakpoint', 'bytearray', 'bytes', 'callable', 'chr', 'classmethod', 'compile', 'complex', 'copyright', 'credits', 'delattr', 'dict', 'dir', 'divmod', 'enumerate', 'eval', 'exec', 'exit', 'filter', 'float', 'format', 'frozenset', 'getattr', 'globals', 'hasattr', 'hash', 'help', 'hex', 'id', 'input', 'int', 'isinstance', 'issubclass', 'iter', 'len', 'license', 'list', 'locals', 'map', 'max', 'memoryview', 'min', 'next', 'object', 'oct', 'open', 'ord', 'pow', 'print', 'property', 'quit', 'range', 'repr', 'reversed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super', 'tuple', 'type', 'vars', 'zip']
>>>
```

```
>>> type(print)
<class 'builtin_function_or_method'>
>>> def hello( ): # hello 함수 정의
        print('Hello')


>>> type(hello)
<class 'function'>
```

내장 함수(Built-in Function)

- 리스트에 사용할 수 있는 내장함수
- len(), max(), min(), sum(), sorted() 함수

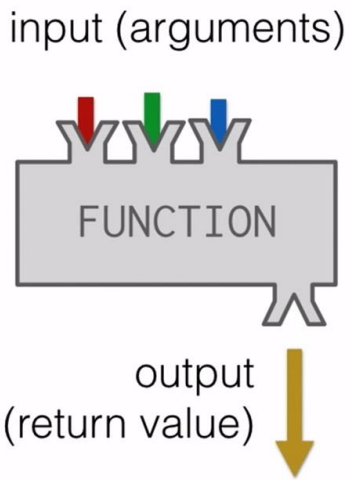
```
>>> L = [10, 5, 8 ,2 ,7, 4, 3, 1, 20]
>>> len(L)  # 리스트의 항목의 개수
9
>>> min(L)  # 리스트의 항목 중 가장 작은 값
1
>>> max(L)  # 리스트의 항목 중 가장 큰 값
20
>>> sum(L)  # 리스트의 항목들의 합
60
>>> sorted(L)  # 리스트의 항목들의 오름차순으로 정렬
[1, 2, 3, 4, 5, 7, 8, 10, 20]
```

함수의 기본구조

```
square(side) # 정수를 제곱하는 함수
compute_average(pointList) # 평균을 구하는 함수
set_cursor_type(c)  # 커서의 타입을 설정하는 함수
```

- def는 함수를 만들 때 사용되는 예약어 (함수를 정의하는 키워드)
- 함수명 : 변수명 규칙과 동일, 함수 목적을 설명하는 동사 또는 동사 + 명사를 사용
- 함수명 뒤에 괄호 안에는 외부에서 전달되는 데이터를 함수로 전달하는 매개 변수 목록을 ‘,’로 구분하여 기술, 생략 가능
- 괄호 뒤에는 반드시 ‘:’를 기술하여 함수 정의가 끝나지 않음을 표시
- 이후 함수의 내용을 기술하고 마지막으로 ‘return’ 예약어와 함께 반환될 값을 기술한다. 반환값이 없을 경우 생략 가능
- 함수는 정의되었다고 해서 바로 실행되지 않는다. 함수가 호출될 때 실행된다.

```
#함수 정의
def 함수명(매개변수1, 매개변수2, …, 매개변수n):
        함수내 수행할 문장1
        함수내 수행할 문장 2
        …
        return 반환값

#함수 호출
함수명(인수1, … , 인수n)
```



함수 작성 예시

```
>>> def say_hello(name): # 함수 정의
        print('안녕, ', name)


>>> say_hello('철수')
안녕,  철수
>>>
>>> def say_msg(name, msg):
        print('안녕', name, '야', msg)


>>> say_msg('철수', '어서 집에 오너라')
안녕 철수 야 어서 집에 오너라
```

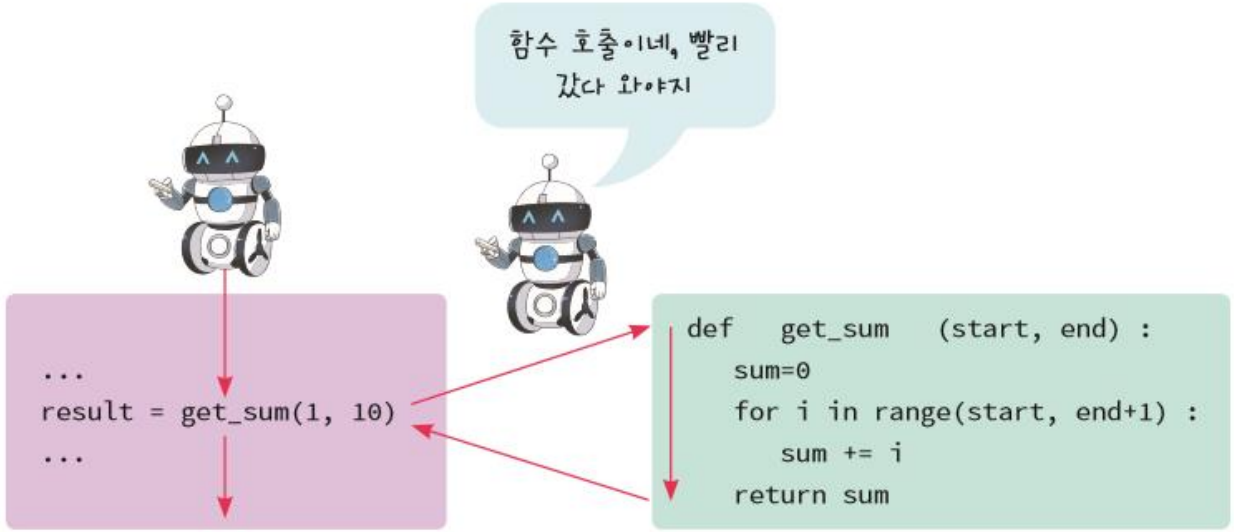
```
def get_sum(start, end) :  #함수 정의
    total = 0
    for i in range(start, end+1) :
        total += i
    return total

value = get_sum(1, 10) #함수 호출
print(value)
print(get_sum(1, 100)) #함수 호출
```

함수 사용의 장점

- **프로그램 안에서 중복된 코드를 제거**
- **복잡한 프로그래밍 작업을 더 간단한 작업들로 분해할 수 있다.**
- **함수의 재 사용성**
  - 자주 이용되는 기능을 작은 크기의 코드로 작성한 후 필요할 때마다 호출하여 사용함으로써 큰 프로그램을 보다 손쉽게 작성할 수 있도록 해준다.
  - 함수 내에서 어떠한 일이 이루어지는지 프로그래머는 알 필요없이 호출만 하면 그 결과를 얻을 수 있기 때문에 보다 손쉬운 프로그램이 가능
- **함수를 사용하면 가독성이 증대되고, 유지 관리도 쉬워진다.**

함수 호출

- **함수를 사용하려면 함수를 호출(call)하여야 한다.**
  - 함수 안의 문장들은 호출되기 던까지는 전혀 실행되지 않는다. 함수가 호출되면 함수 안에 있는 문장들이 순차적으로 실행되며, 실행이 끝나면 호출한 위치로 되돌아간다.



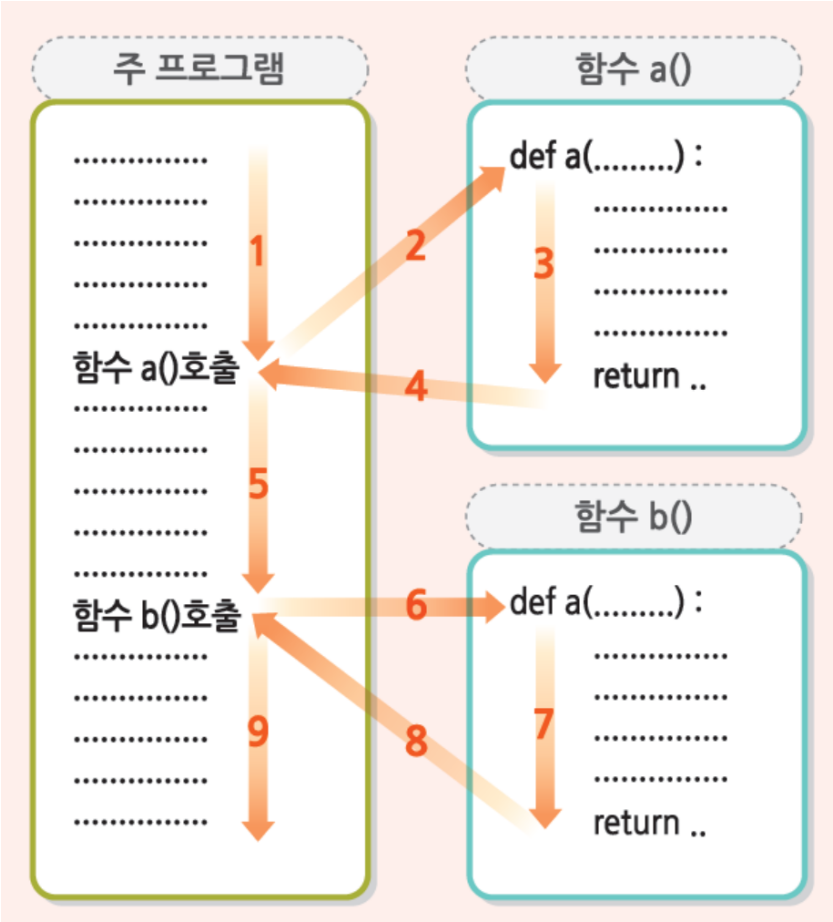
함수 호출

- 함수는 여러 번 호출할 수 있다.

```
...
x = get_sum(1, 10)
...
y = get_sum(1, 20)
```

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

프로그램 내에서 함수를 호출하는 관계

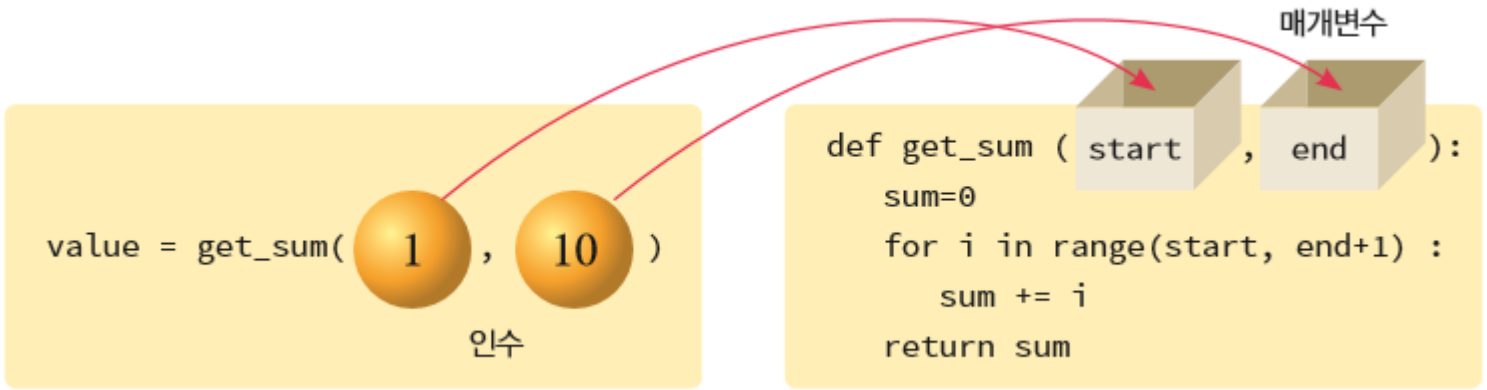


함수 작성과 호출 예시

```
#함수 정의
def findMax(a, b): #a, b: 매개변수(parameter)
    if a > b :
        fmax  = a
    else :
        fmax = b
    return fmax

#main

#함수 호출
num1 = findMax(10, 20)  # 10, 20 : 인수(arguments)
num2 = findMax(5, 9) # 5, 9 : 인수(arguments)

print(num1, num2)
```

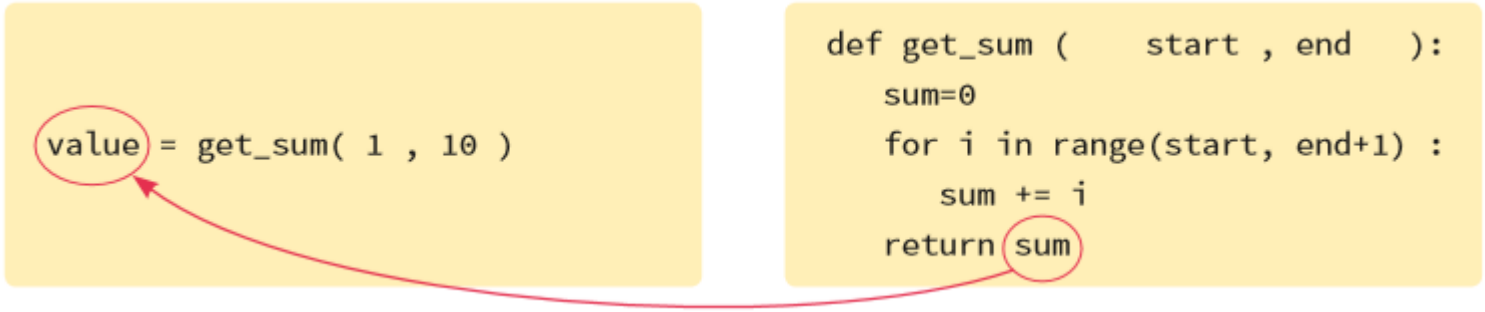
함수에 입력 전달하기(인수와 매개변수)

- 인수(argument) : 호출 프로그램에 의하여 함수에 실제로 전달되는 값(정보)
- 매개 변수(parameter)는 이 값을 전달받는 변수
- 인수가 여러 개인 함수 : 인수의 개수만큼 parameter 필요
- 함수의 인수는 개수에 상관없이 모든 데이터 타입의 인수를 취할수 있다.



값 반환하기

- 함수는 값을 반환할 수 있다.
- 반환값(return value)은 함수가 호출한 곳으로 반환하는 작업의 결과값이다.



값 반환하기

함수에서 return 문이 여러 번 나오는 경우

- return 문이 여러 번 나오더라도 가장 먼저 return을 만나는 순간 함수는 값을 반환하고 종료한다.
- return 키워드 : 함수를 실행했던 위치로 돌아가게 함.
  만약 리턴 뒤에 데이터가 있으면 그 데이터를 가지고 돌아감.

```
def findMax(a, b):
    if a > b :
        fmax  = a
    else :
        fmax = b
    return fmax
```

```
def findMax(a, b):
    if a > b :
        return a
    else :
        return b
```

값 반환하기

- return문 다음에 반환값을 지정하지 않는 경우
  ➔ None 라는 특수한 값 반환
- None : 어떠한 객체도 참조하지 않는다

```
#함수 정의
def test( ) :
    return 100

#main
value = test() #함수 호출
print(value)
```

```
100
>>>
```

```
#함수 정의
def test( ) :
    return

#main
value = test() #함수 호출
print(value)
```

```
None
>>>
```

```
#함수 정의
def test(msg):
    s = msg

#main
print(test('Hello')) #함수호출
```

```
None
>>>
```

yunyi96@gmail.com

함수의 위치

- 함수는 호출 전에 정의되어 있어야 한다.

```
print('프로그램의 시작')
hello()
hello()
print('프로그램의 끝')

def hello():
    print('Hello Word!')
```

```
프로그램의 시작
Traceback (most recent call last):
  File "C:/Users/yuni/Desktop/7장/ex_hello.py", line 2, in <module>
    hello()
NameError: name 'hello' is not defined
```

```
def main():
    print('프로그램의 시작')
    hello()
    hello()
    print('프로그램 종료')

def hello():
    print('Hello Word!')

main()
```

```
프로그램의 시작
Hello Word!
Hello Word!
프로그램 종료
```

```
print('test함수 정의 전')

def test( ) :
    print('테스트입니다.')

print('함수 호출 시작')
test()
print('함수 호출 끝')
```

```
test함수 정의 전
함수 호출 시작
테스트입니다.
함수 호출 끝
```

Docstring (Document String – 문서화)

도움말

- 함수에 대한 설명을 함수 내에 넣을 수 있는 기능으로, 함수의 기능, 모든 인수, return값, 일어나는 예외 등을 문서화한다.
- Docstring 작성방법 : def 문 다음 첫번째 줄에 삼중 따옴표를 사용하여 작성

```
def greet(name):
    '''
    This function greets to the person passed in as a parameter
    '''
    print("Hello, " + name + ". Good morning!")
```

Docstring (Document String – 문서화)

- Docstring을 출력하는 방법
  - docstring은 해당 객체 의 __doc__ 특수 속성이 된다.
  - __doc__ 속성을 이용하거나 help 함수 사용

```
>>> def greet(name):
        ''' This function greets to the person passed in as a parameter '''
        print("Hello, " + name + ". Good morning!")


>>> help(greet)
Help on function greet in module __main__:

greet(name)
    This function greets to the person passed in as a parameter

>>> print(greet.__doc__)
 This function greets to the person passed in as a parameter
```

```
>>> print(print.__doc__)
print(value, ..., sep=' ', end='₩n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file:  a file-like object (stream); defaults to the current sys.stdout.
sep:   string inserted between values, default a space.
end:   string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.
>>> help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='₩n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

Lab 온도 변환 함수 (파일명 temp_converter.py)

- 섭씨 온도를 화씨 온도로 변환하여 반환하는 함수 FtoC()를 작성하고 테스트하라.

Lab: 소수 찾기 (파일명 prime.py)

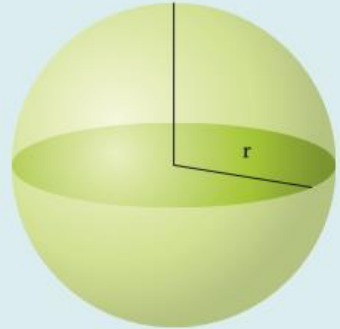
- 여기서는 소수를 판별하는 함수 is_prime()을 작성하여 사용하여 보자.

```
정수를 입력하시오: 101
True
```

Lab: 구의 부피 계산하기

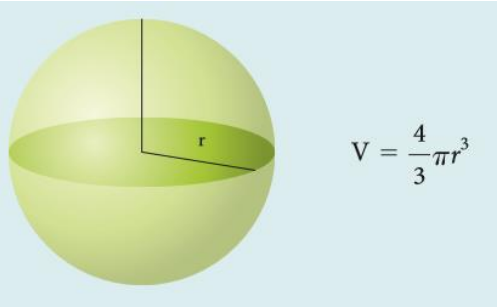
- 구의 부피를 계산하는 함수 sphereVolume()을 작성하여 보자. 반지름이 r인 구의 부피는 다음과 같다.

```
구의 반지름을 입력하시오: 10.0
4188.790204786391
```



$$V = \frac{4}{3}\pi r^3$$

Solution: 파일명 sphere.py



Lab: 패스워드 생성기

- 일회용 패스워드 생성기를 이용하여서 3개의 패스워드를 생성하여 출력하는 프로그램을 작성해보자.

```
5bvn2u
bdhq31
ynaefv
```

Solution : 파일명 password.py

```
>>> import random
>>> help(random.choice)
Help on method choice in module random:

choice(seq) method of random.Random instance
    Choose a random element from a non-empty sequence.
```

yunyi96@gmail.com

디폴트 인수

- 파이썬에서는 함수의 매개변수가 기본값을 가질 수 있다. 이것을 디폴트 인수(default argument)라고 한다.

```
def calc_tax(price, tax_rate):
    total = price + (price * tax_rate)
    return total

totalPrice = calc_tax(10000,0.2)
print("세율 적용시 가격 : ", totalPrice)
```

```
세율 적용시 가격 :  12000.0
>>>
```

```
def calc_tax(price, tax_rate=0.1):
    total = price + (price * tax_rate)
    return total

totalPrice = calc_tax(10000)
print("세율 적용시 가격 : ", totalPrice)
```

```
세율 적용시 가격 :  11000.0
>>>
```

함수 인수를 보내는 방법

- 위치로 매칭하는 방법: 위치 인수(positional argument)
  인수와 매개변수에 위치와 일치시키는 인수
- 키워드로 매칭하는 방법: 키워드 인수(keyword argument)
  매개변수에 이름을 명시적으로 지정해서 전달하는 인수

```
def calc(x, y, z):
        return x-y+z
```

```
def sumsub(a, b, c=0, d=0):
    return a - b + c - d

print(sumsub(12, 4))
print(sumsub(42, 15, d=10))
print(sumsub(42, 15, 10))
```

```
>>> calc(y=20, x=10, z=30)
20
```

```
>>> calc(20, 10, 30)
40
```

```
>>> calc(20, y=10, z=30)
20
```

섞일 수 있지만 , 위치 인수가 키워드 인수 앞에 나와야 한다.

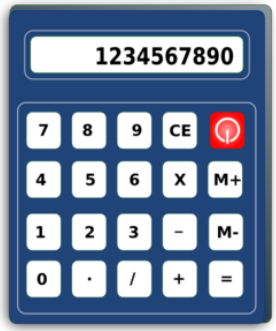
```
>>> calc(x=20, 10, 30)
```

SyntaxError: positional argument follows keyword argument

Lab: 패스워드 생성기

- 사칙 연산을 수행하는 4개의 함수(add(), sub(), mul(), div())를 작성한다. 이들 함수를 이용하여 10+20*30을 계산하여 보자. 함수를 호출할 때 키워드 인수를 사용하여 호출해보자.

610



yunyi96@gmail.com

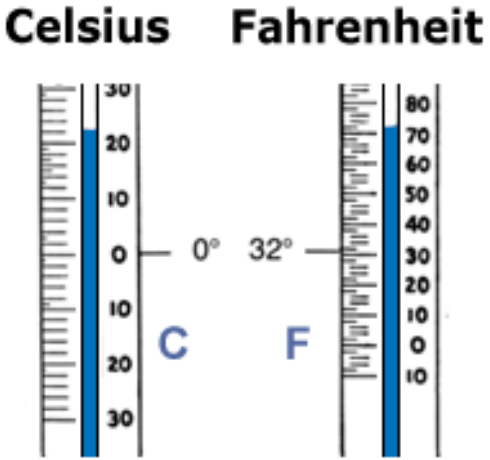
Solution : 파일명 calc.py

```
def add(a, b):
    return a + b

def sub(a, b):
    return a - b

def mul(a, b):
    return a * b

def div(a, b):
    return a / b

r1 = mul(a=20, b=30)
r2 = add(a=10, b=r1)
print(r2)
```

Lab: 온도변환기

- 섭씨 온도를 화씨 온도로, 또 그 반대로 변환하는 프로그램을 작성하여 보자.

```
'c' 섭씨온도에서 화씨온도로 변환
'f' 화씨온도에서 섭씨온도로 변환
'q' 종료
메뉴에서 선택하세요.c
섭씨온도: 100
화씨온도: 212.0
'c' 섭씨온도에서 화씨온도로 변환
'f' 화씨온도에서 섭씨온도로 변환
'q' 종료
메뉴에서 선택하세요.f
화씨온도: 100
섭씨온도: 37.77777777777778
'c' 섭씨온도에서 화씨온도로 변환
'f' 화씨온도에서 섭씨온도로 변환
'q' 종료
메뉴에서 선택하세요.q
```



yunyi96@gmail.com

Solution : 파일명 temp.py

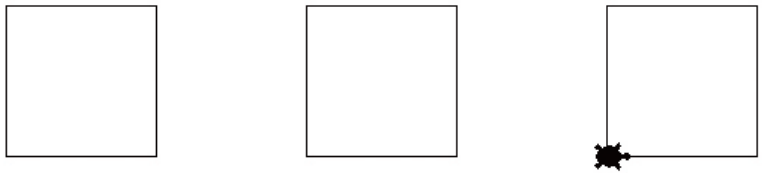
yunyi96@gmail.com

예제: 사각형을 그리는 함수 작성하기

- 정사각형을 그리는 함수는 다음과 같다.

```
def square(length):      # length는 한변의 길이
        for i in range(4):
                t.forward(length)
                t.left(90)
```

- 위의 함수를 호출하여 3개의 정사각형을 그려 보자.

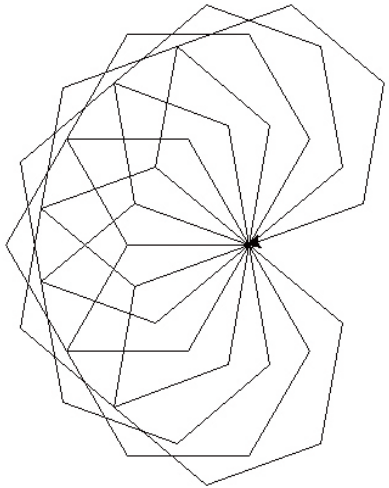


Solution

```
import turtle
t = turtle.Turtle()
t.shape("turtle")

def square(length): # length는 한변의 길이
        for i in range(4):
                t.forward(length)
                t.left(90)

t.up()                  # 펜을 든다.
t.goto(-200, 0)         # (-200, 0)으로 이동한다.
t.down()                # 펜을 내린다.
square(100);            # square() 함수를 호출한다.
t.up()
t.goto(0, 0)
t.down()
square(100);
t.up()
t.goto(200, 0)
t.down()
square(100);
```

Lab: n-각형을 그리는 함수 작성하기

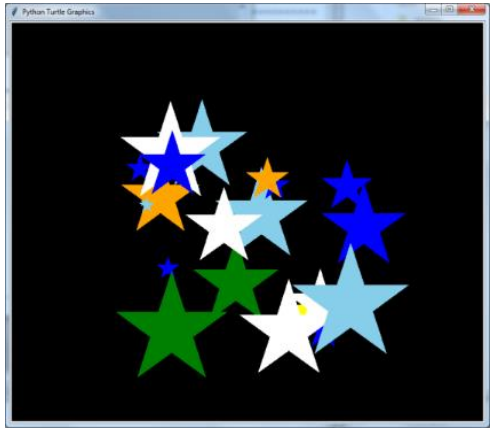
- n각형을 그리는 함수를 작성한 후 왼쪽으로 20도씩 회전하면서 10개의 n각형을 함수를 이용하여 그린다.



```
import turtle
t = turtle.Turtle()

# n-각형을 그리는 함수를 정의한다.
def n_polygon(n, length):
        for i in range(n):
                t.forward(length)
                t.left(360//n)

for i in range(10):
    t.left(20)
    n_polygon(6, 100)
```

Lab 별 그리기

- **별을 랜덤한 위치에 그리는 프로그램을 작성하시오.**
  - 화면의 (x,y) 위치에 별을 그리는 함수 작성
    draw_star(color, length, x, y)



Solution : 파일명 randomStar.py