

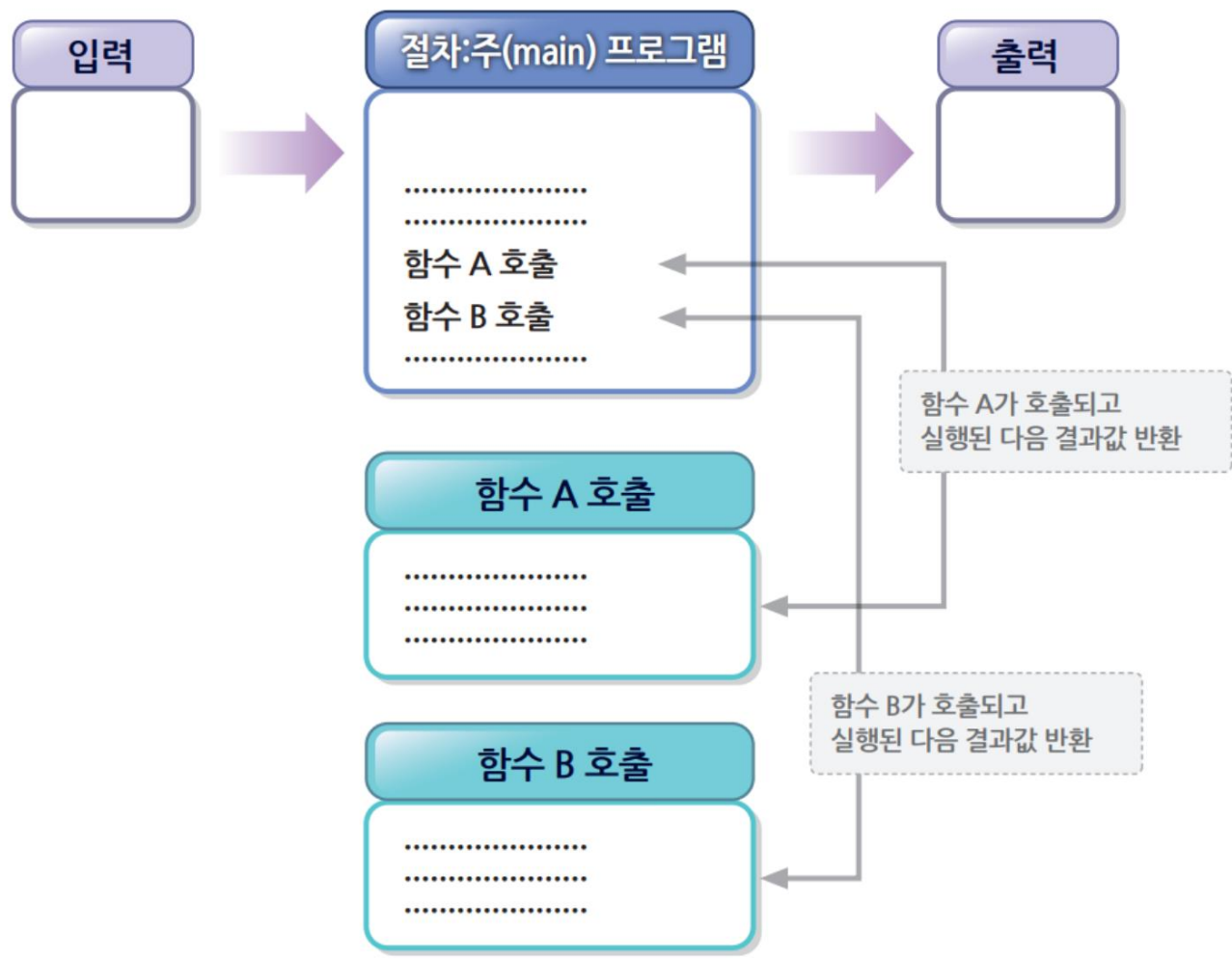


# 함수와 모듈

# Section 1 프로그램과 함수

- 프로그램?
  - 일상생활의 문제를 컴퓨터로 해결하는 과정을 기술한 절차
- 프로그램을 하나로 작성?
  - 프로그램의 논리가 복잡하고 대규모 프로그램인 경우에 프로그램을 하나로 작성한다는 것은 매우 어려우며 많은 문제점을 내포
- 함수function ?, 부 프로그램sub program ?, 프로시저procedure ?
  - 프로그램의 규모가 커질 경우, 프로그램을 작은 단위로 나누어 여러 개의 프로그램으로 작성하는 것이 원칙

Section 1 프로그램과 함수



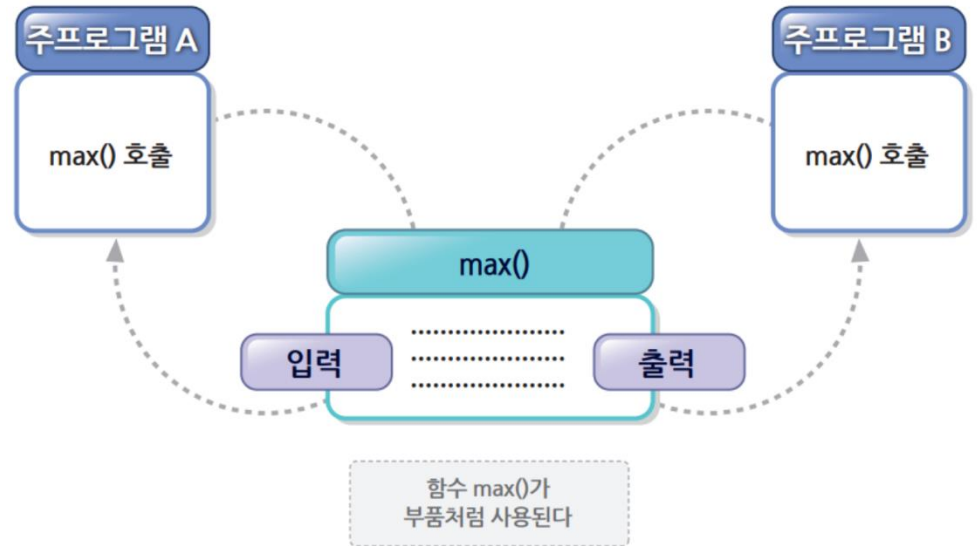
## Section 2 함수 Function

- 함수 function
  - 서브루틴subroutine, 프로시저procedure, 부 프로세스sub process
  - 정의 : 특정 처리를 실행하여 결과를 반환하는 프로그램
- 특정 기능을 수행하는 함수를 만들어 놓고, 서로 다른 프로그램에서 부품처럼 가져다 사용

## Section 2 함수 Function

### • 함수 사용 장점

- 하나의 프로그램에서 반복 사용되는 부분을 함수로 작성함으로써, 코드의 크기를 줄이고 재사용
- 프로그램을 기능 중심으로 부품화 함으로서, 이해가 쉽고 유지보수가 용이
- 핵심 기능을 부품으로 제공함으로서, 다른 프로그램에서도 재사용 가능



## Section 2 함수 Function

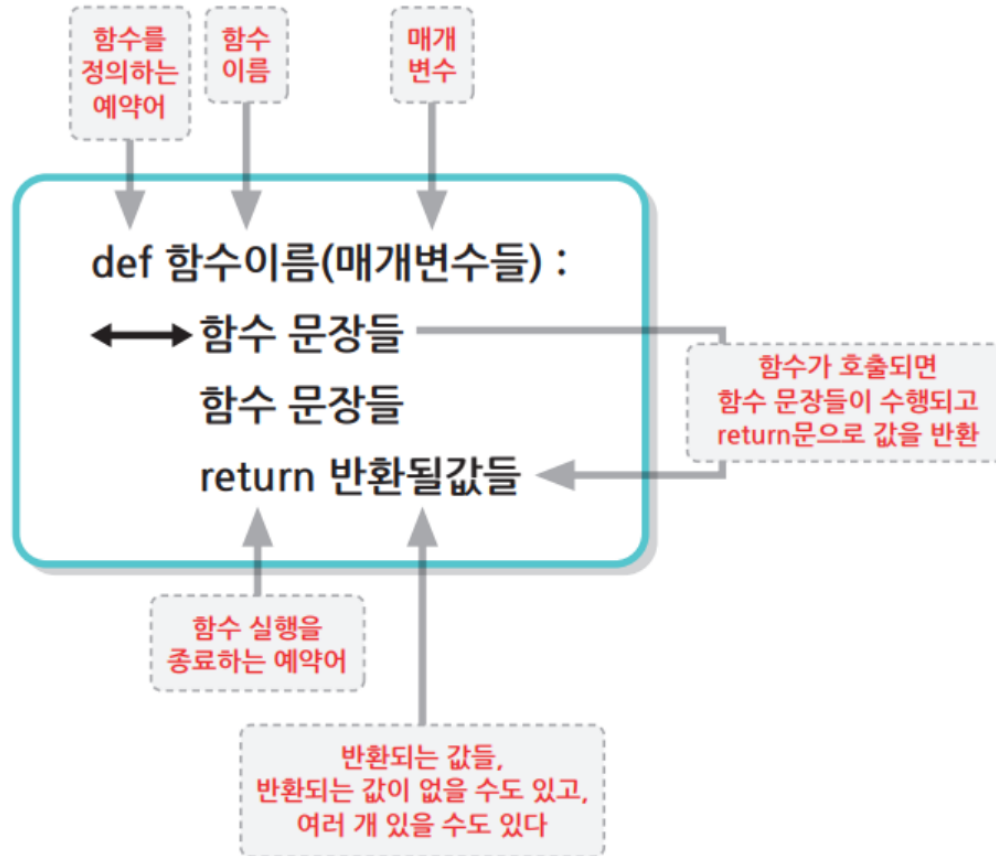
---

- 파이썬 함수

- 내장 함수 : 프로그램에서 많이 사용되는 기능을 함수로 미리 만들어 놓은 것
- 사용자 정의 함수 : 사용자가 프로그램을 작성하면서 필요로 되는 특정 기능을 함수로 작성하여 사용하는 것을 의미

# 1. 함수의 정의와 함수 호출

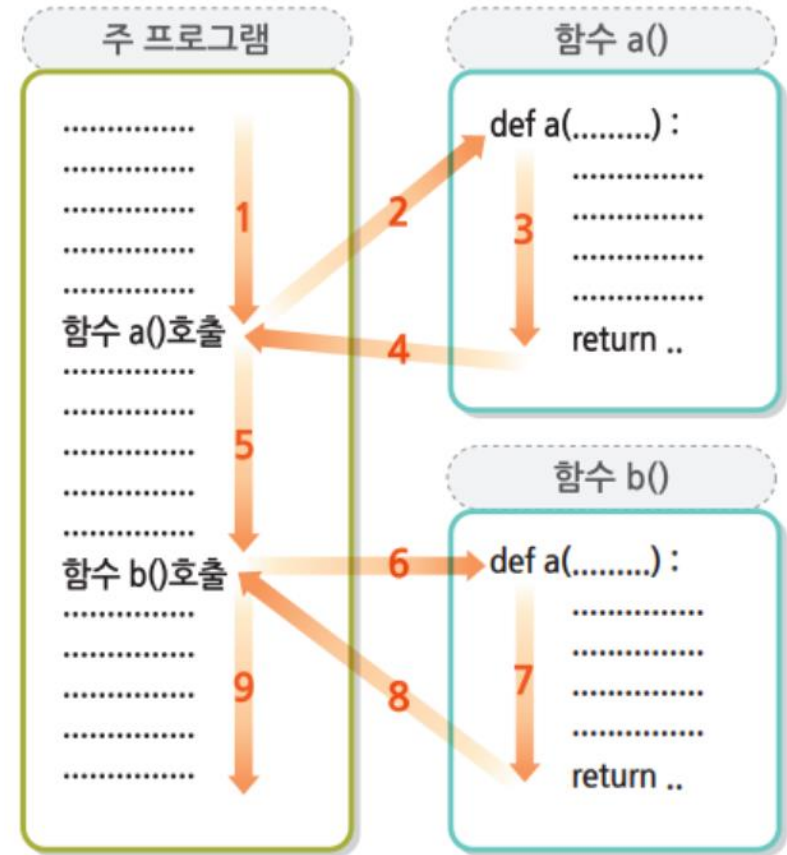
## ■ 함수의 정의



# 1. 함수의 정의와 함수 호출

## ■ 함수 정의

- "def 함수이름(매개변수)"를 사용
- def는 예약어이며, 매개변수는 없을 수도 있지만, 없는 경우에도 반드시 괄호는 있어야 함
- 함수가 반환 값이 있을 경우에는 "return"문을 사용하여 값을 반환





# 1. 함수의 정의와 함수 호출

```
def BigSmall(a,b) :           # 함수 정의(함수가 호출되기 전에 정의)
    if a > b :
        big=a                 # 함수의 몸체
        small=b
    else :
        big=b
        small=a
    return big,small          # 함수 결과의 반환

a=int(input("첫 번째 숫자 입력 : "))
b=int(input("두 번째 숫자 입력 : "))

x,y=BigSmall(a,b)             # 함수를 호출하고 결과 값을 변수에 저장

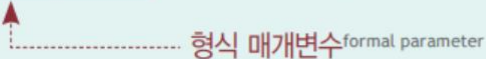
print("큰 값 :",x)
print("작은 값 :",y)
```

```
===== RESTART: D:/program/chap08/8-2-1-1.py =====
첫 번째 숫자 입력 : 100
두 번째 숫자 입력 : 200
큰 값 : 200
작은 값 : 100
```

## 2. 함수의 매개변수

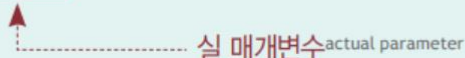
- 매개변수 : 함수를 호출할 때 넘겨주는 값을 저장하는 변수
  - 함수 호출할 때 변수 : 실 매개변수 actual parameter
  - 함수 호출 당할 때 변수 : 형식 매개변수 formal parameter

```
def volumeBox(width, height, depth) : # 함수 정의
    #.....
    #.....
    #.....
```



```
w=int(input("가로를 입력 : "))
d=int(input("세로를 입력 : "))
h=int(input("높이를 입력 : "))
```

```
volume=volumeBox(w, d, h) # 함수 호출
#.....
```



## 2. 함수의 매개변수 ① 위치에 의한 값 전달

- 함수에 값을 전달 할 때, 값의 매칭이 위치에 따라 진행되는 방법

```
def volumeBox(width, height, depth) : # 함수 정의
#.....
#.....
#.....

w=int(input("가로를 입력 : "))
d=int(input("세로를 입력 : "))
h=int(input("높이를 입력 : "))

volume=volumeBox(w, d, h) # 함수 호출

#.....
```

실 매개변수와 형식 매개변수의  
위치에 따라서 각각 매칭

## 2. 함수의 매개변수 ②묵시적default 값을 가진 매개변수

- 함수의 형식 매개변수에 묵시적인 값을 지정하여 다양한 형태로 사용이 가능

```
def volumeBox(width=1, height=1, depth=1) :      # 함수 정의. 형식 매개변수의 묵시적 값을 지정
#.....
#.....
#.....

w=int(input("가로를 입력 : "))    실 매개변수와 형식 매개변수의
d=int(input("세로를 입력 : "))    위치에 따라서 각각 매칭
h=int(input("높이를 입력 : "))

volume1=volumeBox()              # 묵시적 값으로 함수 호출
volume2=volumeBox(w)
volume3=volumeBox(w,d)           # 1~3개의 값으로 함수 호출. 지정되지 않은 값은 묵시적 값을 사용
volume4=volumeBox(w,d,h)
#.....
```

묵시적 값을 지정

묵시적 값을 이용한 다양한 호출

## 2. 함수의 매개변수 ③ 이름에 의한 값 전달

- 함수에 값을 전달할 때 이름 정보를 지정하여 전달

```
def volumeBox(width, height, depth) :  # 함수 정의
    #.....
    #.....
    #.....

#.....
volume1=volumeBox(depth=20, height=10, width=30)  # 순서와 상관없이 이름을 지정하여 호출
volume2=volumeBox(height=10, width=30, depth=20)
#.....
```

### 3. 함수의 반환 값



- 함수는 호출되면 지정된 기능을 수행하고, 수행된 결과를 반환
  - `return`문을 사용하여 함수를 종료하고 값을 반환
  - 함수에 `return`문이 없거나, 단순히 `return`문만 사용되는 경우에는 반환 값이 없는 함수

# 3. 함수의 반환 값

```
def morning1(name) :           # 함수 정의. 반환값이 없다
    print(name, "씨 굿모닝입니다")

def morning2(name) :           # return 값이 한개인 경우
    s=name+"씨 굿모닝입니다"
    return s

def morning3(name) :           # return 값이 여러개인 경우
    s1="오늘 날씨가 쾌청합니다"
    s2=name+"씨 굿모닝입니다"
    return s1,s2

#.....
morning1("파이썬")             # 반환값이 없는 함수 호출

s=morning2("파이썬")           # 반환값이 한 개 있는 함수 호출
print(s)

s1, s2=morning3("파이썬")      # 반환값이 두 개 있는 함수 호출. 두 개의 반환값이 차례로 s1, s2에 배정
print(s1,s2)
#.....
```

## 4. 함수의 매개변수 자료형



- 파이썬에서 변경 가능한 자료
  - 리스트, 집합, 사전 자료형
- 변경 가능한 자료형을 함수의 매개변수로 사용할 경우
  - 함수에서 그 값을 변경하는 경우 호출한 프로그램에서도 해당되는 매개변수의 값이 변경되는 부작용(side effect)이 발생





## 5. 함수에서의 지역변수Local variable와 전역변수Global variable



- 지역변수 : 함수 내에서 사용하는 변수(매개변수 포함)
  - 함수에서 선언된 지역변수는 함수 내에서만 유효
- 전역변수 : 메인 프로그램에서 선언되거나, 함수에서 global로 선언된 변수
  - 이러한 전역변수는 메인이나 함수에서 자유롭게 사용 가능



```
def func1(a,b) :
    c=a+b
    print("x,y의 값 :",x,y)

def func2(x,y) :
    x=x*2
    y=y*2
    print("x,y의 값 :",x,y)
    global z
    z=x+y

x=10
y=20
func1(x,y)
#print(c)

func2(x,y)
print(x,y)
print(z)
```

# a,b는 지역변수  
# c 지역변수  
# 메인에서 선언된 변수는 global 변수. x,y값으로 10과 20 출력

# global 변수와 동일한 이름을 사용. x,y는 지역변수  
# 지역변수 x,y 값으로 20과 40 출력  
# global 변수 선언  
# 두 값의 합을 global 변수에 저장

# 함수 호출  
# 함수에서 지역변수로 선언된 c값 출력?. 오류발생

# 함수 호출  
# 메인에서 선언된 변수의 값 10, 20 출력  
# 함수에서 선언된 global 변수 z 값 60 출력

# 실습 1

사용자로부터 3개의 숫자를 입력받아 가장 큰 수를 구하는 프로그램을 작성하시오. 3개의 숫자를 매개변수로 받아서 가장 큰 수를 반환하는 부분을 findMax(a,b,c) 함수로 작성하시오.

## 실행결과

```
===== RESTART: D:/program/chap08/예제8-2-6-1.py =====  
첫 번째 숫자 입력 : 200  
두 번째 숫자 입력 : 100  
세 번째 숫자 입력 : 170  
가장 큰 수는 : 200
```

# 실습 2



사용자로부터 두 개의 숫자를 입력받아 두 숫자 사이(두 숫자 포함)의 소수<sup>prime number</sup>를 출력하는 프로그램을 작성하시오. 두 개의 숫자를 매개변수로 받아서 두 숫자 사이의 소수를 출력하고, 소수의 개수를 반환하는 부분을 findPrime(x,y) 함수로 작성하시오.

## 실행결과

```
===== RESTART: D:/program/chap08/예제8-2-2.py =====  
첫 번째 숫자 입력(작은 수) : 17  
두 번째 숫자 입력(큰 수) : 71  
17 부터 71 사이의 소수 : 17 19 23 29 31 37 41 43 47 53 59 61 67 71  
소수는 모두 14 개 입니다
```

# Section 3 모듈



- 모듈

- 서로 관련 있는 프로그램들을 모아놓은 것으로 정의
- 모듈에는 함수, 클래스, 데이터들이 포함될 수 있다

- 파이썬 모듈

- 사용자가 직접 생성하여 사용하는 사용자 생성 모듈
- 파이썬 패키지 안에 포함된 표준 모듈
- 회사 등에서 특정 기능을 제공하는 서드파티Third Party 모듈

# 1. 사용자 생성 모듈 ① 모듈의 생성

## ■ 사용자 생성 모듈

# polyArea.py 파일로 모듈 생성

pi = 3.14159

# 함수에서 사용할 상수 선언

def rectangleArea(width, depth) :  
 return width\*depth

# 사각형의 넓이를 계산하는 함수

def triangleArea(base, height) :  
 return base\*height/2

# 삼각형의 넓이를 계산하는 함수

def circleArea(r) :  
 return pi\*r\*r

# 원의 넓이를 계산하는 함수

def circumference(r) :  
 return 2\*pi\*r

# 원의 둘레를 계산하는 함수

# 1. 사용자 생성 모듈 ② 모듈 사용을 위한 선언과 패스 설정

## ■ 모듈 사용

- 프로그램의 처음에 import문을 사용하여 원하는 모듈의 사용을 정의
- 모듈의 사용을 정의하기 위해서는 사용하려는 모듈이 있는 곳의 경로를 패스로 지정

```
>>> import polyArea                                # 모듈이 저장된 곳에 패스가 설정되지 않아 오류 발생
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    import polyArea                                # 모듈이 저장된 곳에 패스가 설정되지 않아 오류 발생
ModuleNotFoundError: No module named 'polyArea'

>>>
>>> import sys                                    # 시스템 표준 모듈 import
>>> sys.path.append("d:\Wprogram\Wchap08")         # 모듈이 저장된 폴더를 패스에 추가
>>>
>>> import polyArea                                # 모듈 import
>>> dir(polyArea)                                  # 모듈의 속성과 함수를 나타낸다
['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'circleArea', 'circu
mference', 'pi', 'rectangleArea', 'triangleArea']
```

- W를 /로 변경해야 함.
- vscode에서 ctrl+shift+p후 인터프리터에서 지정



# 1. 사용자 생성 모듈 ③ 모듈의 사용

## ■ 모듈의 사용

### – 모듈에서 선언된 함수를 사용하기 위해서는 모듈과 함수 이름을 같이 사용

```
import polyArea                                # 모듈 import

print("사각형의 넓이를 계산합니다")
width=float(input("사각형의 가로 : "))
depth=float(input("사각형의 세로 : "))
print("사각형의 넓이는 :", polyArea.rectangleArea(width,depth))    # 모듈의 함수를 호출

print("삼각형의 넓이를 계산합니다")
base=float(input("삼각형의 밑변 : "))
height=float(input("삼각형의 높이 : "))
print("삼각형의 넓이는 :", polyArea.triangleArea(base,height))    # 모듈의 함수를 호출

print("원의 넓이와 둘레를 계산합니다")
r=float(input("반지름 : "))
print("원의 넓이는 :", polyArea.circleArea(r))                    # 모듈의 함수를 호출
print("원의 둘레는 :", polyArea.circumference(r))                # 모듈의 함수를 호출

# print("파이 값 :", pi)    # 오류 발생
print("파이 값 :", polyArea.pi)    # 모듈의 이름과 같이 사용
```

# 1. 사용자 생성 모듈 ③ 모듈의 사용

- 모듈에서 특정 요소만을 사용

- import문을 from문과 같이 사용
- 모듈명 없이 함수 이름으로 직접 사용 가능

```
from polyArea import pi, circleArea    # 모듈 polyArea에서 pi, circleArea import

print("파이 값 :", pi)                  # 모듈명 없이 pi값 직접 사용
# print("파이 값 :", polyArea.pi)      # 모듈명을 사용하는 경우 오류 발생

print("원의 넓이와 둘레를 계산합니다")
r=float(input("반지름 : "))
# print("원의 넓이는 :", polyArea.circleArea(r))  # 오류 발생
print("원의 넓이는 :", circleArea(r))        # 함수명을 직접 사용
```

## 2. 표준 모듈 ①random 모듈



- 주로 난수와 연관된 기능들을 모아 놓은 모듈
  - 정수와 연관된 함수 : randint(a,b)와 randrange(a,b,c)

```
>>> import random                # random 모듈 import
>>>
>>> print(random.randint(1,100))  # 1~100 사이의 정수 중에서 임의의 수를 반환
98
>>>
>>> print(random.randrange(10))   # 0~9 사이의 정수 중에서 임의의 수를 반환
4
>>> print(random.randrange(1,100)) # 1~100 사이의 정수 중에서 임의의 수를 반환
7
>>> print(random.randrange(1,100,2)) # 1~100 사이의 홀수 중에서 임의의 수를 반환
61
```

## 2. 표준 모듈 ①random 모듈



- 실수와 연관된 함수 : `random()`, `uniform(a,b)`

```
>>> import random                # random 모듈 import
>>>
>>> print(random.random())        # 0.0 이상 1.0 미만의 실수를 반환
0.13362725309072254
>>>
>>> print(random.uniform(100.1,100.2))  # 100.1 이상 100.2 미만의 실수를 반환
100.19124857579718
```

## 2. 표준 모듈 ①random 모듈



- 문자열, 리스트, 튜플 등과 같이 자료구조에 적용할 수 있는 random 모듈 :  
**choice(x), shuffle(x)**

```
>>> import random                # random 모듈 import
>>>
>>> s='Python'
>>> print(random.choice(s))       # 문자열에서 임의의 문자 반환
h
>>>
>>> color=['red','green','blue']
>>> print(random.choice(color))   # 리스트에서 임의의 항목을 반환
green
>>>
>>> num=(1,2,3,4,5)
>>> print(random.choice(num))     # 튜플에서 임의의 항목을 반환
4
>>>
>>> list1=[1,2,3,4,5]
>>> print(list1)                 # 리스트 출력
[1, 2, 3, 4, 5]
>>> random.shuffle(list1)        # shuffle() 함수를 이용하여 리스트의 요소를 섞어서 list1에 저장
>>> print(list1)                 # 리스트 출력
[3, 2, 1, 4, 5]
```

## 2. 표준 모듈 ②itertools 모듈

- 반복적인 처리를 하는데 유용한 함수들을 가지고 있는 모듈
  - 반복되는 요소들을 연결하는 `chain()` 함수
  - 무한 반복을 제공하는 `count()` 함수

```
>>> import itertools                # itertools 모듈 import
>>>
>>> list1=[1,2,3]
>>> list2=[4,5,6]
>>> for i in itertools.chain(list1,list2):    # list1, list2를 연결해서 처리
        print(i, end=' ')

1 2 3 4 5 6
>>>
>>> for i in itertools.count(100):          # count() 함수는 숫자를 반복해서 출력. 100부터 무한정 출력
        print(i, end=' ')

100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 1
29 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 Traceback
(most recent call last):
  File "<pyshell#266>", line 2, in <module>
    print(i, end=' ')
KeyboardInterrupt
```

## 2. 표준 모듈 ③copy 모듈

- 파이썬에서 사용되는 다양한 객체들을 복사하는데 많이 사용
  - 1차원 데이터만 복사하는 `copy()` 함수

```
>>> import copy                # copy 모듈 import
>>>
>>> list1=[1,2,3]
>>> list2=list1                # 단순히 리스트를 대입
>>> id(list1),id(list2)        # 두 개의 리스트는 같은 객체
(2991889651400, 2991889651400)
>>> list1[0]=100
>>> list1, list2               # list2도 변경됨
([100, 2, 3], [100, 2, 3])
>>>
>>> list3=[1,2,3]
>>> list4=copy.copy(list3)     # copy() 함수를 사용한 복사
>>> list3[0]=100
>>> list3, list4               # list4는 변경 안됨
([100, 2, 3], [1, 2, 3])
```

# 실습 4

사용자로부터 3개의 숫자를 입력받아 가장 큰 수를 구하는 findMax(a,b,c) 함수, 가장 작은 수를 구하는 findMin(a,b,c), 모든 수의 합을 구하는 findSum(a,b,c) 함수를 find 모듈로 작성하시오. 메인 프로그램에서는 3개의 숫자를 입력받아 모듈의 함수를 호출하도록 실행결과를 참조하여 프로그램을 작성하시오.

## 실행결과

```
===== RESTART: D:/program/chap08/예제8-3-1.py =====  
첫 번째 숫자 입력 : 100  
두 번째 숫자 입력 : 200  
세 번째 숫자 입력 : 300  
가장 큰 수는(findMax()) : 300  
가장 작은 수는(findMin()) : 100  
숫자의 합은(findSum()) : 600
```



# 실습문제



1. 사용자로부터 두 개의 정수를 입력받아 두 숫자 사이의 정수 합(두 숫자 포함)을 구하여 출력하는 프로그램을 작성하시오. 단 두 숫자의 합을 구하는 부분을 함수로 작성하시오.
2. 사용자로부터 하나의 정수를 입력 받아 정수에 해당하는 구구단과 구구단 결과의 합을 구하는 프로그램을 작성하시오. 단 구구단은 함수에서 출력하고, 구구단 결과의 합은 함수에서 반환 받아 메인프로그램에서 출력되도록 작성하시오.
3. 사용자로부터 하나의 정수를 입력받아 그 숫자가 소수인지 판별하는 프로그램을 작성하시오. 단 소수를 판별하는 부분은 함수로 작성하시오.