

파이썬 기초

# 07 함수



홍필두 교수  
(파이썬기초)



## 학습내용

- 01 함수와 인수
- 02 인수의 형식
- 03 함수의 특이사항





## 학습목표

- 함수와 함수에서 사용되는 인수를 이해하고 사용할 수 있다.
- 함수에서 사용되는 인수의 형식을 이해하고 사용할 수 있다.
- 함수내 특징적인 사항을 이해하고 사용할 수 있다.



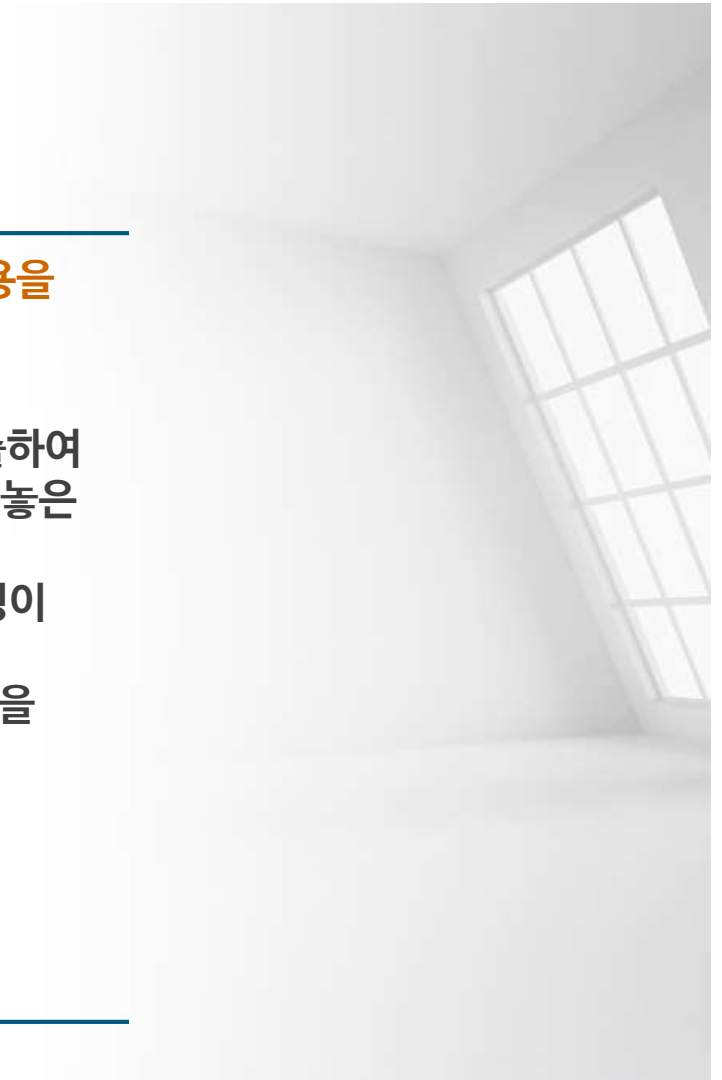
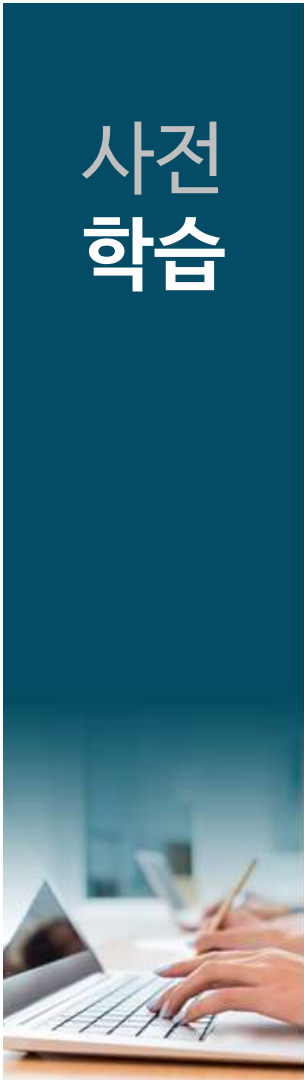
# 사전 학습

## “편리한 함수 사용”

함수는 프로그램을 모아서 정의한 후 해당 내용을 프로그램 내부에서 필요시 호출하여 사용하는 프로그램 기법입니다.

함수는 개발자가 스스로 정의하고 필요시 호출하여 사용할 수도 있고, 이미 다른 사람들이 개발해 놓은 유용한 함수를 편리하게 가져다 사용할 수도 있습니다. 함수를 사용함으로써 프로그램 코딩이 간결해지고, 또 이미 누군가가 정의해 놓은 유용한 함수를 사용함으로써 복잡한 프로그램을 쉽게 구현할 수도 있습니다.

C, java 등 모든 프로그래밍 언어에서도 당연히 함수는 기본적으로 사용됩니다. 여러분은 함수의 일반적인 사항에 대하여 검색을 통해 미리 알아보도록 합시다.



# 01

## 함수와 인수

- 1) 함수의 정의
- 2) 인수
- 3) return 값
- 4) pass

## 1) 함수의 정의

### 함수

프로그램 블록을 미리 정의하고 프로그램 내부에서 호출하여 사용하는 기법



비슷한 로직(블록)이 반복되거나 프로그램 흐름을 구조적으로 표현하기 위하여 사용함



누군가 이미 정의해 놓은 함수를 가져다 마치 블록을 조립하듯이 프로그램을 완성함

```
def 함수(인수 목록):  
    실행내용블록
```

## 1) 함수의 정의



다음 프로그램은 유사한 내용이 반복되는 경우가 많음

```
print("별10개 찍기")
for i in range(10):
    print("*",end="")
print("\n")

print("별3개 찍기")
for i in range(3):
    print("*",end="")
print("\n")

print("별5개 찍기")
for i in range(5):
    print("*",end="")
print("\n")
```

```
>>>
== RESTART:...==
별10개 찍기
*****

별3개 찍기
***

별5개 찍기
*****

>>>
```

## 1) 함수의 정의



함수를 활용하여 프로그램을 작성함



프로그램이 간결해지고 이해하기 쉬워짐



반복 또는 유사한 내용을 간결하게 표시가 가능함



## 1) 함수의 정의

```
def starprint(n):  
    for i in range(n):  
        print("*",end="")  
    print("\n")  
  
print("별10개 찍기");starprint(10)  
print("별3개 찍기");starprint(3)  
print("별5개 찍기");starprint(5)
```

```
>>>  
== RESTART:...==  
별10개 찍기  
*****  
  
별3개 찍기  
***  
  
별5개 찍기  
*****  
  
>>>
```

## 2) 인수

```
def starprint(n):  
    for i in range(n):  
        print("*",end="")  
    print("\n")  
  
print("별10개 찍기");starprint(10)  
print("별3개 찍기");starprint(3)  
print("별5개 찍기");starprint(5)
```

- 앞서 예제에서 함수의 n값에 대응되는 값을 함수로 전달해 줌으로서 원하는 목적(별을 개수에 맞게 출력)을 수행할 수 있음
- 이를 인수라고 함

```
>>>  
== RESTART:...==  
별10개 찍기  
*****  
  
별3개 찍기  
***  
  
별5개 찍기  
*****  
  
>>>
```

## 2) 인수

```
def intsum(n):  
    sum=0;  
    for i in range(1,n+1,1):  
        sum+=i  
    return sum  
  
print("1~10까지의 합:",intsum(10))  
print("1~1000까지의 합:",intsum(1000))  
print("1~10000까지의 합:",intsum(10000))
```

- 1부터 n까지의 합을 계산하는 함수를 정의하고 사용해 봄
- range를 사용할 때 (초기,최종,증가치)인수의 주의

이때 계산결과를 다시 원 프로그램에게 반환하여야 하는 경우가 생김 : 이때 [return 값]의 형태로 반환

```
>>>  
== RESTART: ...==  
1~10까지의 합: 55  
1~1000까지의 합: 500500  
1~10000까지의 합: 50005000  
>>>
```

## 2) 인수

```
def intsum(x,y):  
    sum=0;  
    for i in range(x,y+1,1):  
        sum+=i  
    return sum
```

```
print("1~10까지의 합:",intsum(1,10))  
print("100~1000까지의 합:",intsum(100,1000))  
print("5000~10000까지의 합:",intsum(5000,10000))
```

- 앞 예제를 조금 변형해 봄 :  
x부터 y까지의 합을 구하는 예제를 수행
- 인수는 한 개 뿐만 아니라 여러 개의 변수를  
나열하여 함수에 전달이 가능함

```
>>>  
== RESTART: ...==  
1~10까지의 합: 55  
1~1000까지의 합: 500500  
1~10000까지의 합: 50005000  
>>>
```

### 3) return 값



return 값을 지정하면 함수의 최종결과를 가진 변수와 같이 취급됨



return 값이 없다면 함수 내 블록을 실행하는 것으로 끝남

### 3) return 값

```
def intsum(x,y):  
    sum=0;  
    for i in range(x,y+1,1):  
        sum+=i  
    return sum  
  
print("1~10까지의 합에 5를 더함:",intsum(1,10)+5)  
i=intsum(100,1000)*33  
print("100~1000까지의 합에 33을 곱함:",i)
```



```
>>>  
== RESTART:.. ==  
1~10까지의 합에 5를 더함: 60  
100~1000까지의 합에 33을 곱함: 16353150  
>>>
```

### 3) return 값

#### return 값이 없는 예제



리턴 값이 없는 함수를 어떤 변수로 값을 받으면  
단순히 값이 없는 None이라는 값이 들어감

```
def intsum(x,y):  
    sum=0;  
    for i in range(x,y+1,1):  
        sum+=i  
    print("함수내부에서 sum값을 출력:",sum)  
  
i=intsum(100,1000)  
print("100~1000까지의 합:",i)  
print("100~1000까지의 합에 33을 곱함:",i))
```

```
>>>  
== RESTART: ...==  
함수내부에서 sum값을 출력: 495550  
100~1000까지의 합: None  
100~1000까지의 합에 33을 곱함: None  
>>>
```

## 4) pass



pass는 아무런 처리가 없는 명령



프로그램을 코딩할 때 우선 함수만 정의해 놓고  
pass 처리 후 추후 내용을 채우면서 개발해 나갈 때  
용이함



## 4) pass

 다음 예제는 트와이스 학급의 성적표를 인쇄하는 과정임

이를 구현해 봄

이름	국어	영어	수학	총점	평균
나연	100	90	100	290	96
정연	100	100	80	280	93
모모	90	100	100	290	96
사나	90	80	90	260	86
지효	100	100	100	300	100
미나	80	70	90	240	80
다현	70	60	100	230	76
채영	80	100	90	270	90
쯔위	100	100	100	300	100
반총점	810	800	850	273	91.0

## 4) pass

 다음 예제는 트와이스 학급의 성적표를 인쇄하는 과정임

이를 구현해 봄

이름	국어	영어	수학	총점	평균
나연	100	90	100	290	96
정연	100	100	80	280	93
모모	90	100	100	290	96
사나	90	80	90	260	86
지효	100	100	100	300	100
미나	80	70	90	240	80
다현	70	60	100	230	76
채영	80	100	90	270	90
쯔위	100	100	100	300	100
반총점	810	800	850	273	91.0

맨 위 타이틀 부분은 내용만 인쇄

- 중간부분은 학생 한명의 성적을 처리하고 있음
- 처리내용은 유사함

마지막 부분은 순수 인쇄는 아니고 점수를 계산 후 인쇄

## 4) pass



먼저 main프로그램영역에서 타이틀인쇄,  
아이템 내용인쇄, 맨 아랫줄 인쇄를 처리하겠다고 구상함



데이터를 정의함



일단 함수내용은 추후 작성하기로 하고 pass처리함

## 4) pass

```
def titleprint():
    pass

def itemprint(name,kortotal,engtotal,mathtotal):
    pass

def tailprint():
    pass

#데이터
cname=["나연","정연","모모","사나","지효","미나","다현","채영","쯔위"]
ckor =[ 100,100,90,90,100,80,70,80,100]
ceng =[ 90,100,100,80,100,70,60,100,100]
cmat =[ 100,80,100,90,100,90,100,90,100]

#여기부터 프로그램
titleprint()
for i in range(9):
    itemprint(cname[i],ckor[i],ceng[i],cmat[i])
tailprint()
```

## 4) pass



이후 titleprint(), itemprint(~), tailprint()를 채우면서 프로그램 코딩을 수행함



하나의 함수만 코딩하고 실행하며 결과가 나왔는지 확인하는 과정을 반복하며 쉽게 코딩을 수행함

```
def titleprint():  
    print("="*50)  
    print("이름\t국어\t영어\t수학\t총점\t평균")  
    print("="*50)  
  
def itemprint(name, kor, eng, mat):  
    print(name, "\t", kor, "\t", eng, "\t", mat, "\t", kor+eng+mat, "\t", (kor+eng+mat)//3)
```

## 4) pass

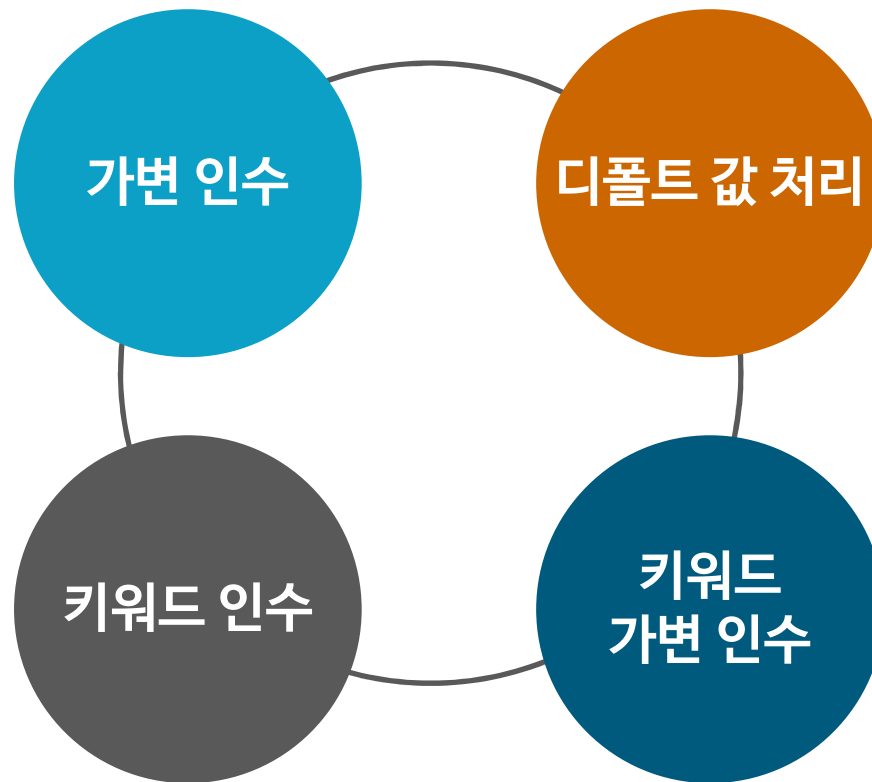
```
def tailprint():
    totkor=0
    for i in ckor:
        totkor+=i
    toteng=0
    for i in ceng:
        toteng+=i
    totmat=0
    for i in cmat:
        totmat+=i
    print("="*50)
    print("반총점\t",totkor,"\t",toteng,"\t",totmat,"\t"\
        ,(totkor+toteng+totmat)//9,"\t",(totkor+toteng+totmat)/3//9)
    print("="*50)
```

# 02

## 변수의 범위

- 1) 인수를 전달할 때 사용할 수 있는 특별한 몇 가지 방법
- 2) 가변인수
- 3) 디폴트 값 처리
- 4) 키워드 인수
- 5) 키워드 가변 인수

## 1) 인수를 전달할 때 사용할 수 있는 특별한 몇 가지 방법





## 2) 가변인수



인수의 개수가 여러 개를 보낼 때 사용



인수 명에 \*을 붙여서 전달

```
def sumsum(*num):  
    sum=0  
    for i in num:  
        sum+=i  
    return sum  
  
print("case 1:", sumsum(1))  
print("case 2:", sumsum(1,2,4))  
print("case 3:", sumsum(1,2,34,567,8901))
```

```
>>>  
== RESTART: ...==  
case 1: 1  
case 2: 7  
case 3: 9505  
>>>
```

## 2) 가변인수



다음의 결과는 앞 예제와 같으나 전혀 다른 방법으로 함수를 정의한 것

```
def sumsum(num):  
    sum=0  
    for i in num:  
        sum+=i  
    return sum
```

- 인수 **num**은 1개이며 해당 인수가 리스트 형이기 때문에 한 개의 변수에 리스트 값을 입력하여 전달
- 별 하나가 큰 차이

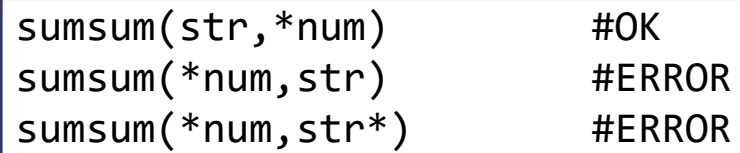
```
print("case 1:", sumsum([1]))  
print("case 2:", sumsum([1,2,4]))  
print("case 3:", sumsum([1,2,34,567,8901]))
```

```
>>>  
== RESTART: ...==  
case 1: 1  
case 2: 7  
case 3: 9505  
>>>
```

## 2) 가변인수

### 주의사항

- 가변인수는 몇 개가 입력될지 모르기 때문에 다음의 경우는 에러가 발생함



sumsum(str, *num)	#OK
sumsum(*num, str)	#ERROR
sumsum(*num, str*)	#ERROR

### 3) 디폴트 값 처리

인수에 디폴트(default)값을 정의하면 해당 인수를  
사용하지 않고 함수의 값을 전달할 경우,  
해당 인수는 디폴트 값을 가지게 됨

### 3) 디폴트 값 처리

#### 합을 구하는 함수 예제



step은 1, end은 1000의 디폴트 값을 지정하였음



인수 하나를 생략하면 step을 기본값으로 처리하며  
두 개를 생략 시 end와 step이 기본값으로 처리됨



defaulttest(start,end=1000,step)은 에러,  
뒤에서 부터 생략이 가능함

### 3) 디폴트 값 처리

#### 합을 구하는 함수 예제

```
def defaulttest (start,end=1000,step=1):  
    sum=0  
    for i in range(start,end+1,step):  
        sum+=i  
    return sum  
  
print("case 1:", defaulttest(0,10,1))  
#인수 하나를 생략시 맨뒤 step값이 1로 기본값처리  
print("case 2:", defaulttest(50,100))  
#인수 하나를 생략시 맨뒤 step값이 1,  
#end 값이 1000으로 기본값처리  
print("case 3:", defaulttest(100))
```

```
>>>  
== RESTART: ...==  
case 1: 55  
case 2: 3825  
case 3: 495550  
>>>
```

## 4) 키워드 인수



함수 호출 시 인수 이름을 명시하여 함수를 실행함



인수 전달 순서를 변경 가능함

## 4) 키워드 인수

```
def defaulttest (start,end=1000,step=1):  
    sum=0  
    for i in range(start,end+1,step):  
        sum+=i  
    return sum  
  
print("case 1:", defaulttest(0,10,1))  
print("case 2:", defaulttest(start=0,end=10,step=1))  
#키워드 지정시 순서가 상관없음  
print("case 3:", defaulttest(step=1,end=10,start=0))  
#맨뒤로 부터만 키워드 인수를 사용하고 앞은 일반인수  
print("case 4:", defaulttest(0,10,step=1))  
#맨뒤로 부터만 키워드 인수를 사용하고 앞은 일반인수  
print("case 5:", defaulttest(0,step=1,end=10))
```

```
>>>  
== RESTART: ...==  
case 1: 55  
case 2: 55  
case 3: 55  
case 4: 55  
case 5: 55  
>>>
```



## 4) 키워드 인수



디폴트 인수와 마찬가지로  
뒤에서부터 키워드 인수를 사용하여야 함



아래와 같은 경우는 에러 발생

```
#print("error1:", defaulttest(0,step=1,10))  
#print("error2:",  
defaulttest(start=0,step=1,10))
```

## 5) 키워드 가변 인수



인수전달 개수가 가변 이며, 변수의 명칭을 키워드로 전달하며 인수 전달 개수도 가변인 경우




함수 호출 시 키워드 인수 방식으로 호출 하여야 함



별을 두 개 (\*\*)로 정의하며,  
다음과 같은 예제의 문법으로 사용함

## 5) 키워드 가변 인수

```
def testfunc (**args):  
    start=args["start"] # 키워드 가변  
    end=args["end"]     # 키워드 가변  
    step=args["step"]   # 키워드 가변  
    sum=0  
    for i in range(start,end+1,step):  
        sum+=i  
    return sum  
  
print("case 1:", testfunc(start=0,end=10,step=1))  
print("case 2:", testfunc(end=100,step=1,start=10))
```



```
>>>  
== RESTART: ...==  
case 1: 55  
case 2: 5005  
>>>
```

## 5) 키워드 가변 인수

```
def testfunc (*s,**args):
    start=args["start"]
    end=args["end"]
    step=args["step"]
    for c in s:
        print(" "*start,end="")
        for i in range(start,end+1,step):
            print(c,end="")
        print("\n")

print("="*30)
testfunc("*","&","=",start=0,end=10,step=1)
print("="*30)
testfunc("*","?","=", "%",start=3,end=10,step=1)
```

- 가변, 키워드 가변을 섞어서 사용하는 것도 가능함

[illegible]

# 03

## 함수의 특이사항

- 1) 전역변수, 지역변수
- 2) docstring

## 1) 전역변수, 지역변수

### 전역변수

- 프로그램 전체에 해당 값이 사용됨

### 지역변수

- 함수에서만 영향력이 미치는 변수

### 주의사항

- C와 Java 와 다르게 파이썬에서는 변수의 선언이 없음
- 함수내에서 값을 대입하는 순간 지역변수가 됨
- 전역변수는 함수에서 해당 값을 바꿀 수 없으며 (바꾸는 경우 지역변수가 됨) 상수로만 사용됨

## 1) 전역변수, 지역변수



함수에서 사용된 지역변수는 다른 함수나 메인 에서 해당 변수를 가져다 사용할 수 없음

```
def test():  
    a=1  
  
test()  
print(a) # 에러발생
```

```
>>>  
== RESTART:  
C:/Users/iamhpd/AppData/Local/Programs/Python/Python37/test1.py ==  
Traceback (most recent call last):  
  File  
    "C:/Users/iamhpd/AppData/Local/Programs/Python/Python37/test1.py",  
    line 5, in <module>  
      print(a) # 에러발생  
NameError: name 'a' is not defined  
>>>
```

## 1) 전역변수, 지역변수

```
def testf1():  
    a=1  
    print("testf1:",a)
```

```
def testf2():  
    a="즐거운 실습"  
    print("testf2:",a)
```

```
def testf3():  
    a=3.141592  
    print("testf3:",a)
```

```
testf1()  
testf2()  
testf3()  
a="랄랄라"  
print(a)
```

testf1(), testf2(), testf3(), main에 선언된 a변수는 이름만 a로 같을 뿐 전혀 다른 변수임

```
>>>  
== RESTART: ...==  
testf1: 1  
testf2: 즐거운 실습  
testf3: 3.141592  
랄랄라  
>>>
```



## 1) 전역변수, 지역변수



전역변수는 메인에서만 변경 가능하며 함수에서는 변경하지 않는 상수로만 사용됨



변경 시 지역변수가 됨



예제 내 case1에서 testf2 함수내 exrate가 지역변수로 testf2에서만 작용하고 함수를 빠져 나와 case2에 testf1을 만나면 기존 지역변수 exrate는 의미가 없이 전역변수 exrate가 사용됨

# 1) 전역변수, 지역변수

```
def testf1():
    print("testf1: 현재 달러환율은 1달러당 ",exrate,"원이며")
    print("testf1: 미화 100달러는 ",100*exrate,"원 입니다")

def testf2():
    exrate=1011
    print("testf2: 현재 달러환율은 1달러당 ",exrate,"원이며")
    print("testf2: 미화 100달러는 ",100*exrate,"원 입니다")

exrate=1021
testf1()
exrate=1120
testf1()

testf2() #case 1
testf1() #case 2
```

```
>>>
== RESTART: ...==
testf1: 현재 달러환율은 1달러당 1021 원이며
testf1: 미화 100달러는 102100 원 입니다
testf1: 현재 달러환율은 1달러당 1120 원이며
testf1: 미화 100달러는 112000 원 입니다
testf2: 현재 달러환율은 1달러당 1011 원이며
testf2: 미화 100달러는 101100 원 입니다
testf1: 현재 달러환율은 1달러당 1120 원이며
testf1: 미화 100달러는 112000 원 입니다
>>>
```

## 2) docstring



함수에 대하여 주석이나 설명을 기입할 때 docstring을 사용



함수 맨 앞에 세 개의 따옴표('\"')로 설명을 시작하고 끝나는 시점에 세 개의 따옴표를 사용



특히 내장 함수나 다른 오픈 라이브러리를 가져다 사용할 때, 사용할 함수가 궁금할 때 대화식 모드에서 help(함수명)을 많이 사용

## 2) docstring

```
def testf1():  
    """  
    이 함수는 홍필두가 최선을 다하여 멋지게 만든 함수  
    인수 값에 숫자만큼 1부터 해당 값을 더한 값을 출력해 준다  
    """  
    sum=0  
    for i in range(n+1):  
        sum+=i  
        print("합:",sum)  
  
# 프로그램에서(특히 대화식 모드) 해당함수가 궁금할 때  
help(testf1)
```

```
>>>  
== RESTART: ...==  
Help on function testf1 in module __main__:  
  
testf1()  
    이 함수는 홍필두가 최선을 다하여 멋지게 만든 함수  
    인수 값에 숫자만큼 1부터 해당 값을 더한 값을 출력해 준다  
  
>>>
```

# 04

## 실습하기 I

- 1) 함수와 인수 - 정의,인수
- 2) 함수와 인수 - return값, pass
- 3) 변수의 범위 - 가변인수

# 실습내용

- 1) 함수와 인수 - 정의,인수
- 2) 함수와 인수 - return값, pass
- 3) 변수의 범위 - 가변인수

# 05

## 실습하기 II

- 1) 변수의 범위 - 디폴트값 처리
- 2) 변수의 범위 - 키워드 인수, 키워드 가변 인수
- 3) 함수의 특이사항 - 전역변수, 지역변수, docstring

# 실습내용

- 1) 변수의 범위 - 디폴트값 처리
- 2) 변수의 범위 - 키워드 인수, 키워드 가변 인수
- 3) 함수의 특이사항 - 전역변수, 지역변수,  
docstring





## 학습활동

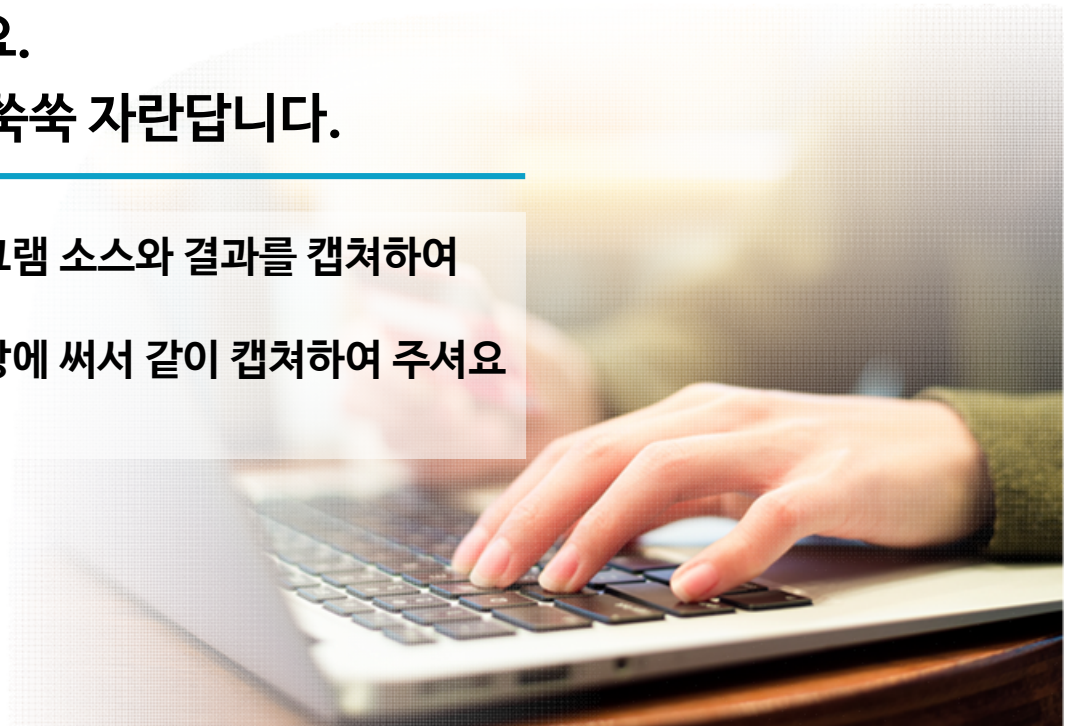
일시정지 버튼을 누른 후, 아래의 학습활동에 참여하세요.

Q

오늘 배운 내용을 스스로 실습하여  
자유게시판에 올려 주세요.

이렇게 정리하면 실력이 쑥쑥 자란답니다.

- ① 본인이 실습한 내용을 프로그램 소스와 결과를 캡처하여 올려주세요
- ② 본인의 학번과 이름을 메모장에 써서 같이 캡처하여 주세요
- ③ 그리고 설명도 달아 주세요





## 학습활동에 대한 교수님 의견

Q

오늘 배운 내용을 스스로 실습하여 자유게시판에 올려 주세요.  
이렇게 정리하면 실력이 쑥쑥 자란답니다.

A

[ 오늘 학습한 내용의 실습 사항 ]

- ① 함수와 인수 - 정의, 인수
- ② 함수와 인수 - return값, pass
- ③ 변수의 범위 - 가변인수
- ④ 변수의 범위 - 디폴트값 처리
- ⑤ 변수의 범위 - 키워드 인수, 키워드 가변 인수
- ⑥ 함수의 특이사항 - 전역변수, 지역변수, docstring

# 학습 평가

Q1

Q2

Q3

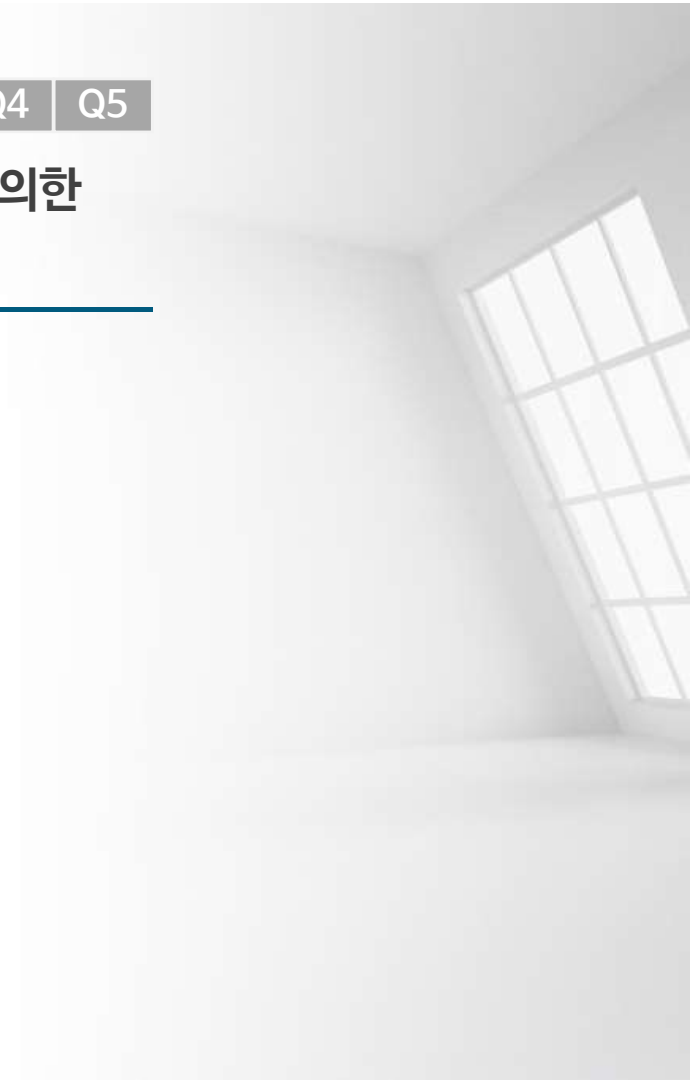
Q4

Q5

Q1

다음 중 함수의 처음시작을 바르게 정의한 것은?

- 1 def test():
- 2 var test():
- 3 func test():
- 4 int test():



# 학습 평가

Q1

Q2

Q3

Q4

Q5

Q1

다음 중 함수의 처음시작을 바르게 정의한 것은?



def test():

2

var test():

3

func test():

4

int test():

정답

1번

해설

def 함수(인수 목록):으로 정의합니다.

# 학습 평가

Q1

Q2

Q3

Q4

Q5

## Q2

함수의 정의가 `def func(x,y,z):`으로 정의되어 있을 때 다음 중 **잘못** 사용된 경우는?

- 1 `func(1,2,3)`
- 2 `a=func(1,2,3)`
- 3 `a=func(1,2,3)+1`
- 4 `func(1,2)`



# 학습 평가

Q1

Q2

Q3

Q4

Q5

Q2

함수의 정의가 `def func(x,y,z):`으로 정의되어 있을 때 다음 중 **잘못** 사용된 경우는?

- 1 `func(1,2,3)`
- 2 `a=func(1,2,3)`
- 3 `a=func(1,2,3)+1`
- ☒ `func(1,2)`

정답

4번

해설

정의된 인수는 전부 전달하여야 하며,  
디폴트 값이 있는 경우에만 생략 가능합니다.

# 학습 평가

Q1

Q2

Q3

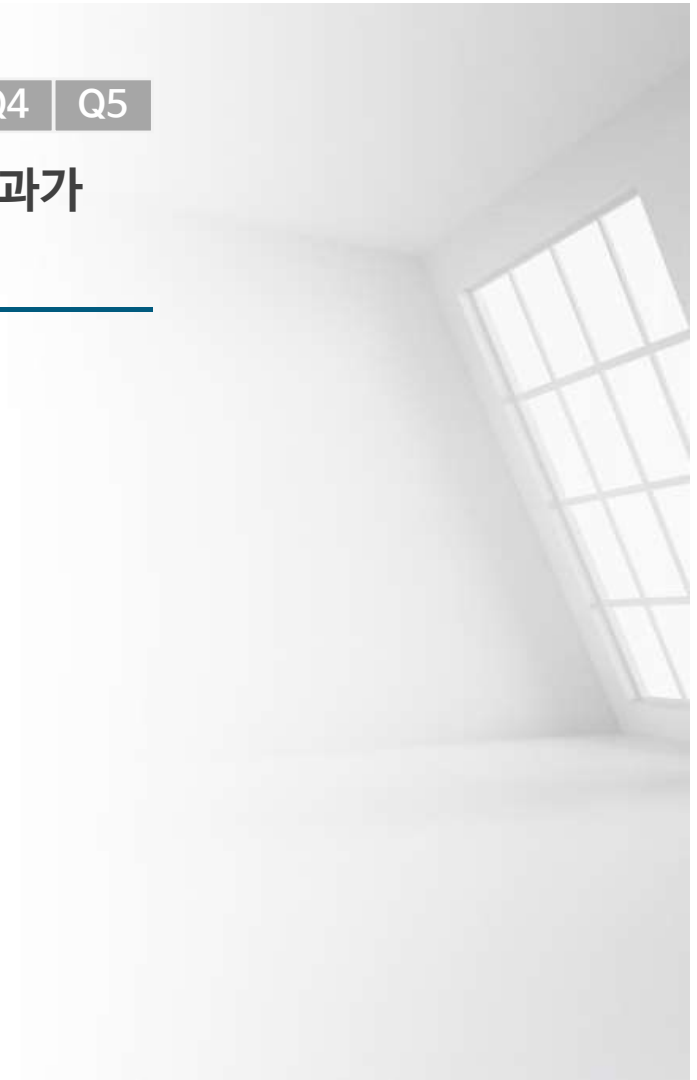
Q4

Q5

## Q3

다음 중 실행을 하여도 특별한 실행결과가 나타나지 **않는** 명령어는?

- 1 break
- 2 pass
- 3 True
- 4 continue



# 학습 평가

Q1

Q2

Q3

Q4

Q5

## Q3

다음 중 실행을 하여도 특별한 실행결과가 나타나지 **않는** 명령어는?

1 break

☒ 2 pass

3 True

4 continue

정답

2번

해설

pass는 아무런 처리가 없는 명령입니다.



# 학습 평가

Q1

Q2

Q3

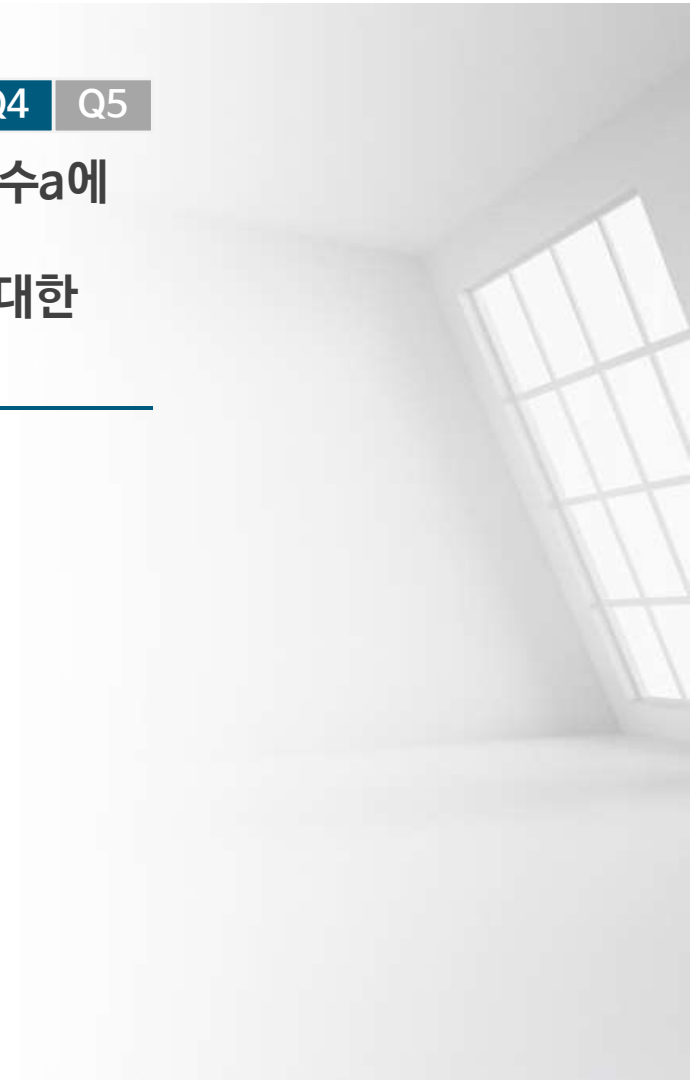
Q4

Q5

## Q4

다음 중 프로그램 문장내에서 같은 함수a에 대하여a(1)로 호출하고 다른 곳에서 a(1,2)라고 호출 된다면 해당함수 a 대한 정의가 바르게 된 경우는 ?

- 1 `def a(*n)`
- 2 `def a(n1,n2)`
- 3 `def a(n1..n2)`
- 4 `def a(**n)`



# 학습 평가

Q1

Q2

Q3

Q4

Q5

## Q4

다음 중 프로그램 문장내에서 같은 함수a에 대하여a(1)로 호출하고 다른 곳에서 a(1,2)라고 호출 된다면 해당함수 a 대한 정의가 바르게 된 경우는 ?



1 def a(\*n)

2 def a(n1,n2)

3 def a(n1..n2)

4 def a(\*\*n)

정답

1번

해설

가변인수를 의미합니다.

# 학습 평가

Q1

Q2

Q3

Q4

Q5

## Q5

다음 중 함수에 대한 설명으로 옳바르지  
**않은** 것은?

- 1 전역변수는 프로그램 전체에 해당 값이 사용된다.
- 2 지역변수는 함수에서만 영향력이 미치는 변수이다.
- 3 전역변수는 함수내에서 변경하더라도  
다시 main함수에서는 해당 값을 유지한다.
- 4 지역변수는 다른 함수에서 사용되지 않는다.

# 학습 평가

Q1

Q2

Q3

Q4

Q5

## Q5

다음 중 함수에 대한 설명으로 옳바르지  
**않은** 것은?

- 1 전역변수는 프로그램 전체에 해당 값이 사용된다.
- 2 지역변수는 함수에서만 영향력이 미치는 변수이다.
- ☒ 3 전역변수는 함수내에서 변경하더라도  
다시 main함수에서는 해당 값을 유지한다.
- 4 지역변수는 다른 함수에서 사용되지 않는다.

정답

3번

해설

C와 Java 와 다르게 파이썬에서는 변수의 선언이 없습니다.  
그러므로 함수내에서 값을 대입하는 순간 지역변수가 됩니다.  
아울러 전역변수는 함수에서 해당 값을 바꿀 수 없으며  
(바꾸는 경우 지역변수가 됨) 상수로만 사용됩니다.



## 정리하기

### 함수와 인수

- ✓ 함수는 프로그램 블록을 미리 정의하고 프로그램 내부에서 호출하여 사용하는 기법을 의미함
- ✓ 인수는 함수 안에 값을 전달해 줌으로서 원하는 목적을 수행할 수 있음
- ✓ **return** 값을 지정하면 함수의 최종결과를 가진 변수와 같이 취급됨
- ✓ **pass**는 아무런 처리가 없는 명령으로 개발작업을 용이하게 하기 위하여 사용





## 정리하기

### 변수의 범위

- ✓ 가변인수는 인수의 개수가 여러 개를 보낼 때 사용하며, 인수에 디폴트(default)값을 정의하면 해당 인수를 사용하지 않고 함수의 값을 전달할 경우, 해당 인수는 디폴트 값을 가지게 됨
- ✓ 함수 호출시 인수 이름을 명시하여 함수를 실행하는 것을 키워드 인수라 하며, 키워드 가변인수는 인수전달 개수가 가변이며, 변수의 명칭을 키워드로 전달하며 인수 전달 개수도 가변인 경우임





## 정리하기

### 함수의 특이사항

- ✓ 파이썬에서는 변수의 선언이 없기 때문에,  
함수 내에서 값을 대입하는 순간 지역변수가 됨
- ✓ 전역변수는 함수에서 해당 값을 바꿀 수 없으며  
(바꾸는 경우 지역변수가 됨) 상수로만 사용됨

