

4. 사용자 정의 함수

함수

□ 내장 함수

- 파이썬에 미리 정의된 함수
- print(), type(), str(), len() 등
- 함수를 제대로 사용하기 위해서는 각 함수 사용하는 방법(input값 설정 등)을 이해해야함

□ 사용자 정의 함수

- 프로그래밍 코드를 작성하다보면 반복되는 코드를 줄여주기 위해 특정 코드를 함수 안에 정의하고, 그 코드를 함수명칭을 호출함으로써 코드를 실행하게 함
- 함수의 구조
 - def(define)함수 사용

Python	R
<pre>def 함수명(파라미터): 실행될 코드 return 결과 값</pre>	<pre>name <- function(args){ statement statement ... return(value) }</pre> <p>Function name: An identifier by which the function is called</p> <p>Arguments: Contains a list of values passed to the function</p> <p>Function body: This is executed each time the function is called</p> <p>Return value: Ends function call & sends data back to the program</p>

함수의 활용

□ 사용자 정의 함수 만들기

- 사용자 정의 함수 활용 예

```

In [1]: #함수

In [4]: def square(x): #제곱
        return x*x

In [5]: square(5)
Out[5]: 25

In [12]: def even_or_odd(n): #홀수 짝수 구분
        if n % 2 == 0:
            print("even")
            return
        print("odd")

In [13]: even_or_odd(3)
odd

In [14]: even_or_odd(4)
even

```

Python	R
<pre> In [18]: def Function_1(a,b): x = a+5 y = b*3 z = x+y return z In [19]: Function_1(1,5) Out[19]: 21 In [20]: Function_1(a=1,b=5) Out[20]: 21 </pre>	<pre> > Function_1 <- function (a,b) { + x <- a+5 + y <- b*3 + z <- x+y + return(z) + } > Function_1(1,5) [1] 21 > Function_1(a=1,b=5) [1] 21 </pre>

- 입력값이 몇 개일지 모를 때
 - 매개변수 앞에 *을 추가
 - def 함수이름(*매개변수):
<수행할 문장>
...
 - *args 매개변수 외에 다른 매개변수도 추가 가능

```
In [21]: #입력변수가 몇개일지 모를때
```

```
In [24]: def add_many(*args):  
    result = 0  
    for i in args:  
        result = result + i  
    return result
```

```
In [25]: add_many(1,2,3)
```

```
Out [25]: 6
```

```
In [26]: add_many(1,2,3,4)
```

```
Out [26]: 10
```

```
In [31]: def add_many(choice,*args): #여러 개의 입력값을 의미하는 *args 매개변수 앞에 choice 매개변수가 추가  
    result = 0  
    if choice == "good":  
        for i in args:  
            result = result + i  
    elif choice == "bad":  
        for i in args:  
            result = result - i  
    return result
```

```
In [32]: add_many("good",1,2,3,4,5,6)
```

```
Out [32]: 21
```

```
In [33]: add_many("bad",1,2,3,4,5,6)
```

```
Out [33]: -21
```

- return을 활용하여 함수 빠져나오기

- 특별한 상황일 때 함수를 빠져나가고 싶다면 **return**을 단독으로 써서 함수를 즉시 빠져나갈 수 있음
- 입력값으로 '바보'라는 값이 들어오면 문자열을 출력하지 않고 함수를 즉시 빠져나감
- 이처럼 return으로 함수를 빠져나가는 방법은 실제 프로그래밍에서 자주 사용됨

```
In [37]: def say_nick(nick):  
    if nick == "바보":  
        return  
    print("나의 별명은 %s 입니다." % nick)
```

```
In [38]: say_nick("바보")
```

```
In [39]: say_nick("천재")
```

```
나의 별명은 천재 입니다.
```

- 매개변수에 초기값 미리 입력하기

- say_myself 함수는 3개의 매개변수를 받아서 마지막 인수인 man이 True이면 "남자입니다.", False이면 "여자입니다."를 출력
- man=True처럼 매개변수에 미리 값을 넣어 줌. 함수의 매개변수에 들어갈 값이 항상 변하는 것이 아닐 경우에는 이렇게 함수의 초기값을 미리 설정해 두면 유용함

```
In [40]: def say_myself(name, old, man=True): #  
         print("나의 이름은 %s 입니다." % name)  
         print("나이는 %d살입니다." % old)  
         if man:  
             print("남자입니다.")  
         else:  
             print("여자입니다.")
```

```
In [41]: say_myself("민주", 41)
```

```
나의 이름은 민주 입니다.  
나이는 41살입니다.  
남자입니다.
```

- 함수 안에 선언한 변수의 효력 범위

- 함수 안에서 새로 만든 매개변수는 **함수 안에서만 사용하는 "함수만의 변수"**임
- 입력값을 전달받는 매개변수 a는 함수 안에서만 사용하는 변수이지 함수 밖의 변수 a가 아님

```
In [42]: #함수 안에서 선언한 변수의 효력 범위  
a = 1
```

```
In [43]: def vartest(a):  
a = a + 1
```

```
In [44]: vartest(a)
```

```
In [45]: print(a)
```

1

- 함수 안에서 함수 밖의 변수를 변경하는 방법

- vartest라는 함수를 사용해서 함수 밖의 변수 a를 1만큼 증가시킬 수 있는 방법
- return 사용하는 방법과 **global 명령어** 사용하여 가능

```
In [50]: a = 1
```

```
In [51]: #return 사용  
def vartest(a):  
a = a + 1  
return a  
  
a = vartest(a)
```

```
In [52]: print(a)
```

2

```
In [56]: a = 1
```

```
In [57]: #global 명령어 사용  
def vartest():  
global a  
a = a + 1
```

```
In [59]: vartest()
```

```
In [60]: print(a)
```

2

