Python 기초 강의

4 Weeks

Cyber-Physical Systems Security

순천향대학교 정보보호학과

김준원

2018.05.07

Agenda

❖함수 (function)

❖사용자 입력

❖출력 결과를 참고한 문제 풀이

함수

함수의 필요성

Python 함수의 구조

익명(Lambda) 함수

함수의 필요성

❖ 함수(Function): 기능, 입력 값에 따라 임의의 출력 값에 대응됨

✓ 예시: $f(x) = 2x + 3$ → $f(2) = 7, f(4) = 11$인 것과 같이 입력 값에 따라 출력 값이 대응됨

❖ 함수의 필요성

✓ 똑같은 내용을 반복해서 작성하고 있다면?

✓ 하나의 코드로 만들어주는 것이 효과적임

✓ 프로그램의 흐름을 일목요연하게 보여줌

✓ 유지보수가 필요할 때 효율적임

✓ 오류 탐지를 하는 데 소요되는 시간이 훨씬 빠름

✓ "함수를 잘 만들 수 있다면 당신은 훌륭한 프로그래머!"

```
함수:
fun(s):
    print("안녕! %s" % s)


함수를 사용하기 전:
print("안녕! %s" % "파이썬")
print("안녕! %s" % "Java")
print("안녕! %s" % "C++")


함수 사용:
fun("파이썬")
fun("Java")
fun("c++")

→ 중복되는 코드를 함수화 시켜서 코드를 정리하는게 훨씬 효율적일 것임
```

Python 함수의 구조

❖ 함수 정의하기

✓ def: 함수를 만들 때 사용하는 예약어

✓ 함수명은 프로그래머가 임의적으로 정할 수 있음

✓ 함수를 정의하여 if, while, for문 등을 같이 사용해 수행할 문장들을 입력해서 완성함

```
def 함수명(매개변수…):
    수행할 문장1
    수행할 문장2
    …
    return 반환할 값
```

❖ 매개변수 : 입력으로 전달된 값을 받는 변수

❖ 인자 : 함수를 호출할 때 전달되는 입력 값

❖ 예시

```
def repeat_print(str, count):
    for i in range(count):
        print(str)
```

→ Count번만큼 문자열 변수 str의 값을 반복 출력하는 함수
→ 매개변수: str, count
→ 인자: count(range 함수 내)

```
def calc(x,y,z):
    return x*x + 2*y + 3*z
```

→ $Calc(x, y, z) = x^2 + 2y + 3z$와 같은 함수
→ 매개변수: x, y, z

Python 함수의 구조

❖ 함수의 형태

✓ 일반적인 함수

```
def 함수명(매개변수):
    수행할 문장1
    수행할 문장2
    …
    return 반환값
```

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

```
>>> a = min(4,6)
>>> a
4
```

✓ 매개변수가 없는 함수

```
def 함수명():
    수행할 문장1
    수행할 문장2
    …
    return 반환값
```

```
def default():
    print("기본값은 2")
    return 2
```

```
>>> a = default()
기본값은 2
>>> a
2
```

Python 함수의 구조

❖ 함수의 형태

✓ 반환 값이 없는 함수

```
def 함수명(매개변수):
    수행할 문장1
    수행할 문장2
    …
```

```
def say(name):
    print("hello %s" % name)
```

```
>>> say("Python")
Hello python
```

✓ 반환 값도 매개 변수도 없는 함수

```
def 함수명():
    수행할 문장1
    수행할 문장2
    …
```

```
def print_test():
    for i in range(4):
        print("Test")
```

```
>>> print_test()
Test
Test
Test
Test
```

Python 함수의 구조

❖ 매개변수의 지정

- ✓ 인자를 순차적으로 넣는다면 각각의 매개변수에 대응됨
- ✓ 매개변수를 지정해서 순서에 상관없이 대입 연산자를 통해 지정할 수 있음

```
예제 함수
def test(a, b):
  …
```

```
>>> test(2, 4)
→ a = 2, b = 4와 같음
```

```
>>> test(4, 2)
→ a = 4, b = 2와 같음
```

```
>>> test(b=4, a=2)
→ 매개변수 지정됨
→ a = 2, b = 4와 같음
```

❖ 가변 인자

- ✓ 입력 값의 개수가 항상 다름
- ✓ 매개변수 앞에 *를 붙여서 사용함
- ✓ 매개변수의 자료형은 튜플

```
예제 함수 (모든 값 더하기)

def sum_total(*args):
    sum = 0
    print(args)
    for i in args:
        sum = sum + i
    return sum
```

```
>>> a = sum_total(1,2,3)
(1,2,3)
>>> a
6
>>> a = sum_total(1,2,3,4,5)
(1,2,3,4,5)
>>> a
15
```

Python 함수의 구조

❖ 매개변수 초기값

- ✓ 매개변수에 대입 연산자를 사용해 초기값을 지정할 수 있음
- ✓ 함수를 호출할 때 해당 매개변수에 대한 인자가 없으면 초기값으로 지정됨
- ✓ 초기에 지정되는 매개변수의 위치는 뒤에 위치시켜야 함
  - ✓ 초기값을 설정해 놓은 매개변수 뒤에 초기값을 설정해 놓지 않은 매개변수를 사용할 수 없음

```
def test(str='Hello world'):
    print(str)
(실제로는 오류 발생, repeat는 printable 앞에 있어야 함)
def multiple(value, printable=True, repeat):
    sum = value
    for i to range(repeat):
        sum = sum * value
        if printable == True:
            print(sum)
    return sum
```

```
>>> test()
Hello world
>>> test('hello')
hello
>>> a = multiple(2, 7)
SyntaxError → 7이 printable의 값인지 repeat
인지 모름
>>> a = multiple(2, false, 3)
>>> a
```

Python 함수의 구조

❖ 키워드 인자

- ✓ 키 값에 대응되는 값을 갖는 인자
- ✓ 매개변수 앞에 **를 붙여서 사용
- ✓ 키워드 매개변수의 자료형은 딕셔너리

예제 함수 (딕셔너리)

```
def result(**kwargs):
    print(kwargs)
    for key in kwargs.keys():
        print("Key: %s, Value: %s" % (key, kwargs[key])
```

```
>>> result(key=1234)
{key=1234}
Key: key, Value: 1234
>>> result(key=1234, name='World', lang='python')
{key=1234, name='world', lang='python'}
Key: key, Value: 1234
Key: name, Value: world
Key: lang, Value: python
```

Python 함수의 구조

❖ return

- ✓ 값을 반환할 때 사용되는 예약어
- ✓ 함수에서의 반환 값은 오로지 1개여야만 함
- ✓ 함수에서 return 키워드를 만나면 함수를 빠져나옴

```
예제 함수 (반환값이 2개)

def double_result(a, b):
    return a*2, b*2
```

```
>>> value = double_result(2,4)
>>> value
(4, 8)
→ 2개 이상이면 튜플형으로 가짐
→ 결론적으로 반환값은 1개
```

```
>>> value, value2 = double_result(2,4)
>>> value
4
>>> value2
8
```

```
예제 함수 (함수 빠져나옴)

def call_name(nickname):
    return
    print("name : %s" % nickname)
```

```
>>> call_name("python")
name: python

→ return 값은 존재하지 않음
→ print 함수를 실행함과 동시에 함수를 빠져 나옴
```

사용자 입력

Input 함수

사용자 입력

❖ 사용자 입력 값

- ✓ 사용자가 입력하는 값을 받고 싶을 때 input() 함수 사용
- ✓ Input 함수 내 인자를 넣으면 입력을 받기 전 문자열을 출력시킴
- ✓ Input 함수에서 받은 인자 값을 대부분 문자열 자료형으로 변수에 넣어 줌

```
>>> a = input()
3 (사용자가 입력한 값)
>>> a
3
```

```
>>> a = input("값을 입력: ")
값을 입력: abc (사용자가 입력함)
>>> a
abc
```

```
>>> a = input()
3 (사용자가 입력한 값)
>>> a
3
```

- ✓ 내장 함수*를 통해 형 변환을 하여 사용할 수 있음

```
>>> a = input()
127
>>> typeof(a)
<class 'str'>
>>> b = int(a)
>>> typeof(b)
<class 'int'>
```

```
>>> a = input()
44+6
>>> print(a)
44+6
```

```
>>> a = input()
Python is best
>>> a
'Python is best'
```

*내장 함수(Internal function): 이미 만들어진 함수로 형 변환 함수의 예로 int, str 함수가 있음

화면 출력을 참고한 문제 풀이

❖ 모든 소수의 합 구하기 (난이도: 중)

- ✓ 소수 : 1과 자기 자신의 값으로만 나누어 떨어지는 수
- ✓ 예를 들어 사용자가 12을 입력했다면, 2부터 12까지 소수를 판별해 소수만 모두 더하면 된다.
- ✓ 따라서 합을 구하는 함수를 $S(x)$라고 하면 $S(12) = 2 + 3 + 5 + 7 + 11 = 28$이라는 결과값이 나온다.
- ✓ 소스 코드 조건
  - ✓ 오른쪽의 출력 결과를 참고하여 소스 코드를 작성하되, 함수 `prime_sum(n)`을 만들어서 소수 값을 합한 값을 반환하는 함수를 구현해주시기 바랍니다.
  - ✓ N의 범위는 $0 \leq N \leq 1000$까지 이며 범위를 벗어나면 오류 메시지가 출력되며 프로그램이 종료됩니다.
  - ✓ 파일명을 prime.py로 해주시기 바랍니다.

```
숫자를 입력하세요: 1
더할 숫자가 없습니다!
숫자를 입력하세요: 0
더할 숫자가 없습니다!
숫자를 입력하세요: 15
2부터 15까지의 모든 소수 합은 41입니다.
숫자를 입력하세요: 21
2부터 21까지의 모든 소수 합은 77입니다.
숫자를 입력하세요: 2
2부터 2까지의 모든 소수 합은 2입니다.
숫자를 입력하세요: -2
범위를 벗어났습니다. 프로그램을 종료합니다.
(프로그램 종료됨)
```

❖ 소스 코드 파일을 ruskonert@gmail.com 또는 support@cpss.network 로 5월 13일까지 보내시기 바랍니다.

화면 출력을 참고한 문제 풀이

❖ 피보나치 값 구하기 (난이도: 중상)

- ✓ 피보나치 : $n-1$번째와 $n-2$번째 값을 더한 값
- ✓ 예를 들어 피보나치 함수를 $f(x)$라고 가정한다면 $f(x)=f(x-1)+f(x-2)$이고 $f(0)=0,\ f(1)=1$이라 둠
- ✓ $f(0)=0,\ f(1)=1,\ f(2)=1,\ f(3)=2,\ f(4)=3,\ f(5)=5,\ f(6)=8$ ...
- ✓ 소스 코드 조건
  - ✓ 오른쪽 출력 결과를 참고하여 소스 코드를 작성하되, 함수 `fibo(n)`을 만들어서 피보나치 함수를 구현해주시기 바랍니다.
  - ✓ n의 범위는 $0 \le N \le 1000$까지 이며 범위를 벗어나면 오류 메시지가 출력됩니다. 이 때, -1을 입력하면 프로그램이 종료되도록 설계하세요.
  - ✓ 재귀 함수로 구현하지 마시기 바랍니다.
  - ✓ 파일명을 fibo.py로 해주시기 바랍니다.

```
숫자를 입력하세요: 10
fibo(10) = 55
숫자를 입력하세요: -5
범위를 벗어났습니다.
숫자를 입력하세요: 0
fibo(0) = 0
숫자를 입력하세요: 1
fibo(1) = 1
숫자를 입력하세요: 4
fibo(5) = 5
숫자를 입력하세요: 7
fibo(7) = 13
숫자를 입력하세요: 1001
범위를 벗어났습니다.
숫자를 입력하세요: -1
프로그램을 종료합니다.
(프로그램 종료됨)
```

❖ 소스 코드 파일을 ruskonert@gmail.com 또는 support@cpss.network 로 5월 13일까지 보내시기 바랍니다.

마침

궁금하신 점이나 질문이 따로 있다면?

개인 이메일 ruskonert@gmail.com

동아리 이메일 support@cpss.network

동아리 저장소 https://github.com/CPSSOpenSource

과제 결과 확인 https://github.com/CPSSOpenSource/Python-Report-Complie

강의를 들어 주셔서 감사합니다.