

공학적 문제해결 기법

9주차 1차시

코드 재사용하기 1





9주차 1차시 : 코드 재사용하기 1

I. 개요

함수를 이용한 코드 재사용하기

II. 학습 개요

1) 학습 목표

함수를 이용하여 코드를 어떻게 재사용하는지 살펴본다. 또한 변수가 함수의 내부나 외부에서 보이도록 할 수 있는 변수의 영역을 관리하는 방법과 def 키워드로 함수를 생성하는 방법도 배운다.

2) 학습 목차(세부 목차)

- 함수
- 변수와 영역
- 키워드 인수와 가변 인수 리스트



9주차 1차시 : 코드 재사용하기 1

III. 학습 1 - 함수

함수 사용하기

- ✓ 어떤 물건을 그냥 버리는 것보다, 가능하다면 다시 사용해야 한다.
- ✓ 프로그래밍 세계에서도 재사용은 매우 중요
- ✓ 재사용은 코드를 더 짧게 할 뿐만 아니라, 더 편하게 읽을 수 있도록 해 준다.
- ✓ 앞에서 코드를 재활용하는 방법들 중에 하나를 이미 살펴보았다.
- ✓ 숫자를 세도록 range와 list 함수이다.
- ✓ 더 많은 숫자들을 가진 리스트를 만든다면, 직접 타이핑하여 생성하는 것은 힘든 일이다.
- ✓ 함수를 사용하면 수천 개의 숫자를 가진 리스트도 쉽게 만들 수 있다.
- ✓ list와 range 함수를 이용하여 많은 숫자들을 가진 리스트를 생성



```
>>> list(range(0, 1000))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 ..., 997, 998, 999]
```



9주차 1차시 : 코드 재사용하기 1

III. 학습 1 - 함수

함수 사용하기

- ☒ 함수: 해야 할 작업에 대한 코드 묶음
- ☒ 프로그램에서 함수를 반복적으로 사용하므로, 코드를 재사용하는 하나의 방법 중 하나
- ☒ 간단하거나 복잡한 프로그램을 만든다면 함수가 유용

함수의 구조

- ☒ 함수는 이름, 매개변수, 내용의 3개의 부분으로 구성

➡ `>>> def testfunc(myname) :`

`print('hello %s' %myname)`

- ☒ 함수의 이름은 testfunc, 하나의 매개변수인 myname을 가짐
- ☒ 매개변수는 함수가 사용될 때만 존재하는 변수



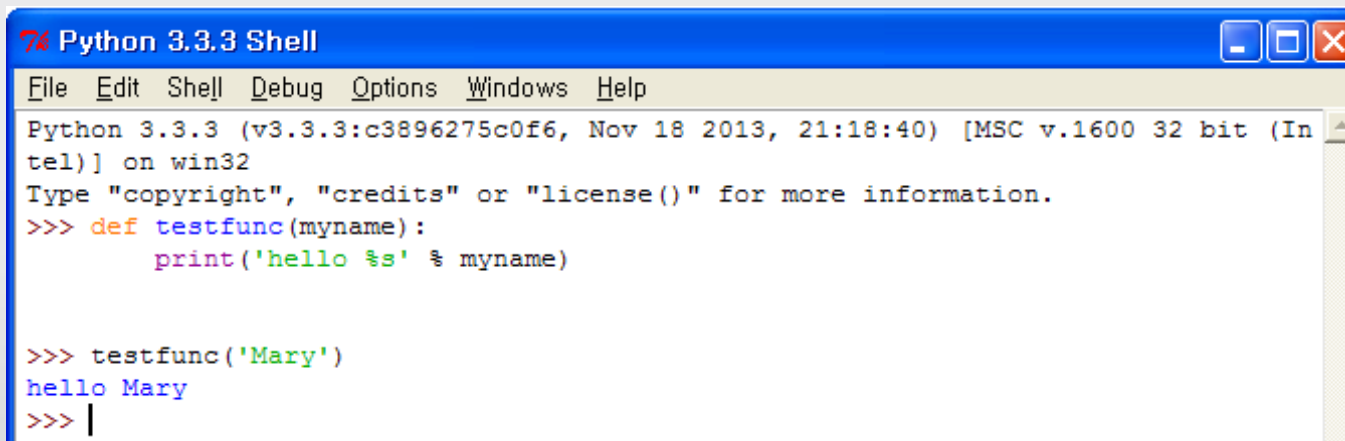
9주차 1차시: 코드 재사용하기 1

III. 학습 1 - 함수

함수 사용하기

 괄호 안의 매개변수 값을 이용하여 함수의 이름을 호출하여 실행

➡ `>>> def testfunc(myname) :`
 `print('hello %s' %myname)`



```
Python 3.3.3 Shell
File Edit Shell Debug Options Windows Help
Python 3.3.3 (v3.3.3:c3896275c0f6, Nov 18 2013, 21:18:40) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> def testfunc(myname):
    print('hello %s' % myname)

>>> testfunc('Mary')
hello Mary
>>> |
```



9주차 1차시 : 코드 재사용하기 1

III. 학습 1 - 함수

함수 사용하기

☒ return 구문을 이용하여 어떤 값을 반환하기 위해 사용 가능

☒ 예를 들어, 얼마나 많은 돈을 저금했는지를 계산하는 함수

➡

```
>>> def savings(pocket_money, paper_route, spending):  
    return pocket_money + paper_route - spending
```

☒ 변수를 먼저 생성한 다음에 함수에서 그것들을 사용할 수도 있다.

➡

```
>>> my_money = savings(10, 10, 5)  
>>> print(my_money)  
>>> print(savings(10, 10, 5))
```

☒ 반환된 결과를 my_money 변수에 저장해서 print문을 이용해 출력한 결과

☒ 반환된 결과를 바로 출력한 결과는 같다.



9주차 1차시 : 코드 재사용하기 1

III. 학습 2 - 변수와 영역

변수와 영역

- ✓ 함수 안에 있는 변수는 그 함수의 실행이 끝나면 더 이상 사용할 수가 없다.
- ✓ 이 변수는 해당 함수 안에서만 존재
- ✓ 프로그래밍 세계에서 이런 것을 영역이라고 부름

➡

```
>>> def variable_test() :  
    first_variable = 10  
    second_variable = 20  
    return first_variable * second_variable
```

- ✓ first_variable과 second_variable을 곱하는 variable_test라는 함수를 생성하고 그 결과값을 반환

➡

```
>>> print(variable_test())  
200
```



9주차 1차시 : 코드 재사용하기 1

III. 학습 2 - 변수와 영역

변수와 영역

- ☒ first_variable(또는 second_variable)의 값을 함수 밖에서 출력하려고 한다면?
- ☒ 다음과 같은 에러메시지를 보게 됨

```
Python 3.3.3 Shell
File Edit Shell Debug Options Windows Help
Python 3.3.3 (v3.3.3:c3896275c0f6, Nov 18 2013, 21:18:40) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> def variable_test():
    first_variable = 10
    second_variable = 20
    return first_variable * second_variable

>>> print(variable_test())
200
>>> print(first_variable)
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    print(first_variable)
NameError: name 'first_variable' is not defined
>>> |
```




9주차 1차시 : 코드 재사용하기 1

III. 학습 2 - 변수와 영역

어떤 변수가 함수 밖에 정의되어 있는 경우

✓ 함수 안에 정의된 변수와는 다른 영역

✓ 함수를 정의하기 전에 변수를 정의하고 함수 안에서 그 변수를 사용해 보자.



```
>>> another_variable = 100
```

```
>>> def variable_test2() :
```

```
    first_variable = 10
```

```
    second_variable = 20
```

```
    return first_variable * second_variable * another_variable
```

✓ first_variable 변수와 second_variable 변수는 함수 밖에서 사용될 수 없다.

✓ another_variable 변수는 함수 밖에서 생성되었기에 함수 안이든 밖이든 사용

✓ 함수를 호출한 결과



```
>>> print(variable_test2())
```

```
20000
```



9주차 1차시 : 코드 재사용하기 1

III. 학습 2 - 변수와 영역

함수를 이용하여 코드를 다시 사용해 보자.

우주선의 동체를 만들기 위해서 매주 2개의 깡통을 평평하게 만들 수 있을 것이라 예상하고, 동체를 완성하는 데 500개의 깡통이 필요하다고 하자. 한 주에 2개의 평평한 깡통을 만들 수 있을 때, 500개의 평평한 깡통을 얻기까지 얼마나 오랜 시간이 걸리는지를 계산하는 함수를 만들어 본다.

- ☒ 일년 동안 매주 얼마나 많은 평평한 깡통들을 만들 수 있는지를 보여주는 함수를 만들어 본다.
- ☒ 매개변수로 깡통의 개수를 받는다.

➡

```
def spaceship_building(cans):  
    total_cans = 0  
    for week in range(1, 53):  
        total_cans = total_cans + cans  
        print('Week %s = %s cans' % (week, total_cans))
```



9주차 1차시 : 코드 재사용하기 1

III. 학습 2 - 변수와 영역

 함수를 이용하여 코드를 다시 사용해 보자.

☒ 여러가지 값들을 넣어 호출해 본다.

➡ `>>> spaceship_building(2)`
`>>> spaceship_building(13)`

☒ 이 함수는 매주 생산되는 강통 개수를 다르게 하여 재사용될 수 있다.

☒ 이것은 다른 값에 대한 결과를 얻기 위해서 매번 for 루프를 다시 타이핑하는 것보다 약간 더 효과적임

```
Python 3.3.3 Shell
File Edit Shell Debug Options Windows Help
Python 3.3.3 (v3.3.3:c3896275c0f6, Nov 18 2013, 21:18:40) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> def spaceship_building(cans):
    total_cans = 0
    for week in range(1,53):
        total_cans = total_cans + cans
        print('Week %s = %s cans' % (week, total_cans))

>>> spaceship_building(2)
Week 1 = 2 cans
Week 2 = 4 cans
Week 3 = 6 cans
Week 4 = 8 cans
Week 5 = 10 cans
Week 6 = 12 cans
Week 7 = 14 cans
Week 8 = 16 cans
Week 9 = 18 cans
Week 10 = 20 cans
Week 11 = 22 cans
Week 12 = 24 cans
Week 13 = 26 cans
Week 14 = 28 cans
```



9주차 1차시 : 코드 재사용하기 1

III. 학습 3 - 키워드 인수와 가변 인수 리스트

키워드 인수

- ☒ 함수 호출에서 키워드 인수란 인수 이름으로 값을 전달하는 방식



```
def area(height, width):
```

```
    return height * width
```

```
a = area(width=20, height=10) # 순서가 아닌 이름으로 값이 전달된다.
```

```
print(a)
```

```
b = area(height='height string ', width=3)
```

```
print(b)
```

- ☒ 출력 결과는 다음과 같다.



```
200
```

```
height string height string height string
```



9주차 1차시 : 코드 재사용하기 1

III. 학습 3 - 키워드 인수와 가변 인수 리스트

키워드 인수 위치

☒ 키워드 인수 전까지는 순서에 의한 인수 매칭

➡ `>>> area(20, width=5)`

☒ 그러나 다음은 가능하지 않다.

➡ `>>> area(width=5, 20)`

☒ 키워드 인수 이후에는 순서에 의한 인수 매칭을 할 수 없기 때문



9주차 1차시 : 코드 재사용하기 1

III. 학습 3 - 키워드 인수와 가변 인수 리스트

가변 인수 리스트

- ✓ 고정되지 않은 수의 인수를 함수에 전달하는 방법
- ✓ 함수를 정의할 때, 인수 리스트에 반드시 넘겨야 하는 고정 인수들을 우선 나열
- ✓ 나머지를 튜플 형식으로 한꺼번

➡

```
>>> def varg(a, *arg):  
    print(a, arg)
```

- ✓ 함수 `varg`를 호출할 때 넘겨지는 첫 인수는 `a`가 받으며 나머지는 모두 튜플 형식으로 `arg`가 받는다.



9주차 1차시 : 코드 재사용하기 1

III. 학습 3 - 키워드 인수와 가변 인수 리스트

가변 인수 리스트

 varg 함수 호출 예

➡ `>>> varg(1)`
1 ()
`>>> varg(2, 3)`
2 (3,)
`>>> varg(2,3,4,5,6)`
2 (3, 4, 5, 6)



『 이 콘텐츠는 2014학년도 학부교육 선도대학 육성사업(ACE)에 의하여 개발한 것임 』

