파이썬 함수

함수란 무엇인가?

쌀, 버터, 올리브, 육수, 향신료 를 넣고 끓이면

리조또가 됩니다

우리는 이미 함수를 배웠습니다

$$y = ax + b$$

여기서 입력은

쌀, 버터, 올리브, 육수, 향신료

x 가 입력 값이 됩니다

그러면 결과는? 리조또, y

이때 리조또는 냄비에 끓이고
함수는 ax+b의 형태로 변화 된다.

파이썬에서의 함수는 def 로 시작 한다.

```
def 함수명(매개변수):
    <수행할 문장1>
    <수행할 문장2>
    ...
```

덧샘 함수를 만들어 봅시다.

```
def add(a, b):
    return a + b
```

매개변수와 인수

```
def add(a, b):  a와 b는 매개변수
    return a + b

add(1, 2)   1, 2는 인수가 됩니다.
```

이처럼 함수는

입력값을 -> 출력값

으로 변환 해준다.

일반적인 함수는

```
def 함수이름(매개변수):
    <수행할 문장>
    ...
    return 결과값
```

입력값이 없는 함수

```
def 함수이름():
    return 결과값
```

결과값이 없는 함수

```
def no_return_add(a, b):
    print(a+b)

no_return_add(1, 2)
```

매개변수 리턴값이 없는 함수

```
def just_hellow():
    print(“just hellow”)

just_hellow()
```

매개변수 리턴값에 따른 함수의 호출

매개O리턴O : 결과값 = function(a, b)
매개X리턴O : 결과값 = function()
매개O리턴X : function(a, b)
매개X리턴X : function()

형태로 호출 한다.

매개변수 지정 호출

```
y = add(a=2, b=3)
print(y)
```

```
y = add(b=3, a=2)
print(y)
```

입력값이 무엇인지 모를때?

```
def 함수이름(*매개변수):
    <수행할 문장>
    ...
```

일반적으로 arg에 s를 붙인 형태를 사용한다.
args
arg는 argument 변수라는 뜻

입력값이 무엇인지 모를때?

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

print(add_many(1,2,3,4,5,6,7,8,9,19))
```

입력값을 두개로 받아서 처리 할때

```
def add_mul(choice, *args):
        if choice == "add":
            result = 0
            for i in args:
                result = result + i
        elif choice == "mul":
            result = 1
            for i in args:
                result = result * i
        return result
```

결과값이 두개인 함수

```
def add_and_mul(a,b):
    return a+b, a*b
```

결과값이 두개인 함수에서 각각의 결과를 따로 보고 싶을때

```
result1, result2 = add_and_mul(1,2)
```

그렇다면 이렇게는?

```
def add_and_mul(a,b):
    return a+b
    return a*b
```

간략한 함수를 이용한 처리 내용

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

지역변수와 전역변수

```
a = 1
def vartest(a):
    a = a +1

vartest(a)
print(a)
```

지역변수와 전역변수

```
a = 1
def vartest(a):
    a = a +1
    return a

a = vartest(a)
print(a)
```

함수안에서 전역변수 변경

```
a = 1
def vartest():
    global a
    a = a+1

vartest()
print(a)
```

권장 하지 않음 나중에 문제가 발생됨

lambda : def와 비슷하나 함수를 한줄로 만든다.

```
def add(a, b):
    return a+b

add = lambda a, b: a+b
```

함수를 사용하는 이유

코드의 재사용을 위해서