데이터 처리를 위한 Python 프로그래밍 입문

8-1강. 함수

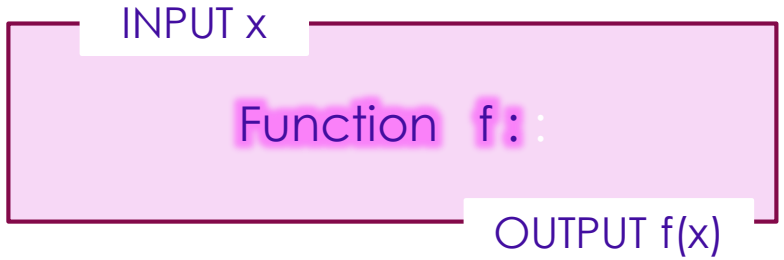
ERICA 2018-2

강의 내용

- 함수 정의 및 호출
- Return문을 통해 값을 반환하는 함수
- Return문이 없는 함수

함수 정의 및 호출

- 함수



- 함수 정의 : 함수가 하는 일을 정의 하는 것
- 함수 호출 : 정의된 함수를 사용하는 것
- 함수 종류
  - return문을 통해 값을 반환하는 함수(Value Returning Functions)
  - return문이 없는 함수(Non-Value Returning Functions)

Return문을 통해 값을 반환하는 함수(1/9)

- Return문이 있는 함수 정의

표기	내 용
def FunctionName(n1, n2, …) : Statements return ResultValue	함수이름이 FunctionName이라는 함수를 정의함 괄호 안의 인자(parameters)인 n1, n2, …을 입력값으로 받아 Statements를 수행하고 그의 결과값인 ResultValue을 반환(return문) 한 후, 함수 종료

Return문을 통해 값을 반환하는 함수(2/9)

- Return문이 있는 함수 정의 시 유의사항

표기	내 용
함수 이름 (Function Name)	• 함수가 수행하는 작업을 요약하는 의미 있는 이름 사용 • 함수 이름 작성시 변수 이름 작성 규칙을 준수 - 맨 처음 단어는 문자로 작성 - 문자, 숫자, _(under bar)로 이루어진 이름을 사용 - 키워드에 있는 이름 사용 불가
인자 (Parameters)	• 인자는 여러 개 사용할 수 있음 • 인자를 사용하지 않을 수 있음 이때는 인자를 생략하고 괄호만 표기
return문	• 결과값을 반환해주는 문장 • return문을 수행하면 함수가 종료됨 • return 뒤에는 값, 수식 또는 변수 등이 사용 될 수 있음 • 값을 반환하지 않고, 작업만 수행하는 함수도 있음 이때는 return문 전부를 생략함

Return문을 통해 값을 반환하는 함수(3/9)

- Return문이 있는 함수 호출

표기	내 용
result = FunctionName()	함수 이름이 FunctionName 이라는 함수를 호출하여 함수가 반환하는 값인 ResultValue를 저장할 변수(result)를 지정하여 저장함.

```
>>> def average(n1, n2):
        result = (n1+n2)/2
        return result

>>>
>>> average(4,11)
7.5
>>>
>>>
>>>
```

Return문을 통해 값을 반환하는 함수(4/9)

- Return문이 있는 함수 호출 : 부피 계산
  - 상황 : 박스(직육면체)의 부피를 구하는 함수를 정의하고, 호출하여 결과 화면과 같이 출력하는 프로그램
  - 결과 화면

```
>>> calculate_volume(3,3,4)
36
>>>
>>> Box_volume = calculate_volume(4,4,6)
>>> print(Box_volume)
96
>>>
```

Return문을 통해 값을 반환하는 함수(5/9)

- Return문이 있는 함수 호출 : 부피 계산
  - 해결 코드

```
def calculate_volume(width, length, height):
    result = width * length * height
    return result
```

Return문을 통해 값을 반환하는 함수(6/9)

- Return문이 있는 함수 호출 : 수학함수
  - 상황 : 수학함수 $y = x^2 + x + 1$ 를 함수로 정의하고, 호출하여 결과 화면과 같이 출력하는 프로그램
  - 결과 화면

```
>>> f(6)
43
>>>
>>> f(3)
13
```

  - 해결 코드

```
...
>>> def f(x):
        result = (x * x) + x + 1
        return result
```

Return문을 통해 값을 반환하는 함수(7/9)

- Return문이 있는 함수 호출 : 수학함수
  - 상황 : int형 정수 1개를 인자로 입력받아, 그 인자보다 3이 큰 값을 반환하는 함수로 정의하고, 호출하여 결과 화면과 같이 출력하는 프로그램
  - 결과 화면

```
>>> a=3
>>> b=6
>>> increase_3(a)
6
>>> increase_3(b)
9
>>>
```

  - 해결 코드

```
>>> def increase_3(num):
        result = num + 3
        return result

>>>
```

Return문을 통해 값을 반환하는 함수(8/9)

- Return문이 있는 함수 호출 : 숫자 list 평균 계산
  - 상황 : list형 숫자로 구성된 list를 인자를 입력받음
    list를 구성하는 element의 평균값을
    결과 화면과 같이 출력하는 프로그램

  - 결과 화면

```
>>>
>>> score=[90,80,100,70]
>>> list_avg(score)
85.0
>>>
```

Return문을 통해 값을 반환하는 함수(9/9)

- Return문이 있는 함수 호출 : 숫자 list 평균 계산
  - 해결 코드

```
def list_avg(numlist):
    sum=0
    avg=0
    for number in numlist:
        sum += number
    avg = sum/len(numlist)

    return avg
```

Return문이 없는 함수(1/6)

- 결과값을 반환하지 않는 함수 정의

표기	내 용
def FunctionName(n1, n2, …) : Statements print ResultValue	함수이름이 FunctionName이라는 함수를 정의함 괄호 안의 인자(parameters)인 n1, n2, …을 입력값으로 받아 Statements를 수행하고 그의 결과값인 ResultValue을 출력(print) 한 후, 함수 종료

```
>>> def average(n1, n2):
        result = (n1+n2)/2
        return result

>>>
>>> average(4,11)
7.5
>>>
```

```
>>> def average(n1, n2):
        result = (n1+n2)/2
        print(result)

>>> average(4,11)
7.5
>>>
>>>
```

Return문이 없는 함수(2/6)

- Return문이 없는 함수 호출 : 수학함수
  - 상황 : 두 수를 곱하는 함수(mul( ))를
    return문 없이 정의하고, 호출하여
    결과 화면과 같이 출력하는 프로그램
  - 결과 화면

```
>>>
>>> mul(3,11)
두 수를 곱한 결과 :  33
>>>
>>>
```

  - 해결 코드

```
...
>>> def mul(n1, n2):
        print('두 수를 곱한 결과 : ', n1 * n2)
```

Return문이 없는 함수(3/6)

- Return문이 없는 함수 호출 : 홀짝
  - 상황 : 정수를 받아 홀수인지 짝수인지를 출력하는 함수를
    return문 없이 정의하고, 호출하여
    결과 화면과 같이 출력하는 프로그램
    * if num %2 == 0
  - 결과 화면

```
>>> is_even_or_odd(3)
3 is odd.
>>> is_even_or_odd(4)
4 is even.
>>>
```

Return문이 없는 함수(3/6)

- Return문이 없는 함수 호출 : 홀짝
  - 상황 : 정수를 받아 홀수인지 짝수인지를 출력하는 함수를
    return문 없이 정의하고, 호출하여
    결과 화면과 같이 출력하는 프로그램
    * if num %2 == 0
  - 해결 코드

```
>>> def is_even_or_odd(num):
        if num%2 == 0:
            print(num, "is even.")
        else:
            print(num, "is odd.")
```

Return문이 없는 함수(4/6)

- Return문이 없는 함수 호출 : 원소 추가
  - 상황 : 정수와 정수 list를 받아 정수가 정수list에 존재여부를 return문 없이 정의하고, 호출하여 결과 화면과 같이 출력하는 프로그램
  - 결과 화면

```
>>> list_add(3,[4,5,6])
원소를 추가하였습니다.
>>> list_add(3,[3,6,9])
이미 있는 원소입니다.
```

Return문이 없는 함수(4/6)

- Return문이 없는 함수 호출 : 원소 추가
  - 상황 : 정수와 정수 list를 받아 정수가 정수list에 존재여부를 return문 없이 정의하고, 호출하여 결과 화면과 같이 출력하는 프로그램
  - 해결 코드

```
def list_add(new_num, num_list):
    if (new_num in num_list):
        print('이미 있는 원소입니다.')
    else:
        num_list=num_list+[new_num,]
        print("원소를 추가하였습니다.")
```

Return문이 없는 함수(5/6)

- Return문이 없는 함수 호출 : 요리 동전 추천
  - 상황 : 동전던지기로 저녁메뉴를 결정
    앞면이 나오면 중국요리, 뒷면이 나오면 일본요리로 결정
    return문 없이 정의하고, 호출하여
    결과 화면과 같이 출력하는 프로그램
  - 결과 화면

```
>>> coin_food('앞')
중국 요리로 결정되었습니다.
>>>
>>> coin_food('뒤')
일본 요리로 결정되었습니다.
>>> coin_food('압')
일본 요리로 결정되었습니다.
>>> coin_food('back')
일본 요리로 결정되었습니다.
>>>
```

함수명은 coin_food()
원소가 '앞' 일 경우 중국요리
그외 의 경우는 모두 일본요리임에 유의합니다.

Return문이 없는 함수(6/6)

- Return문이 없는 함수 호출 : 요리 동전 추천(계속)
  - 해결 코드

```
def coin_food(coin):
    if (coin == "앞"):
        print("중국 요리로 결정되었습니다.")
    else:
        print('일본 요리로 결정되었습니다.')
```

Thank you