KG 아이티뱅크

파이썬

PYTHON

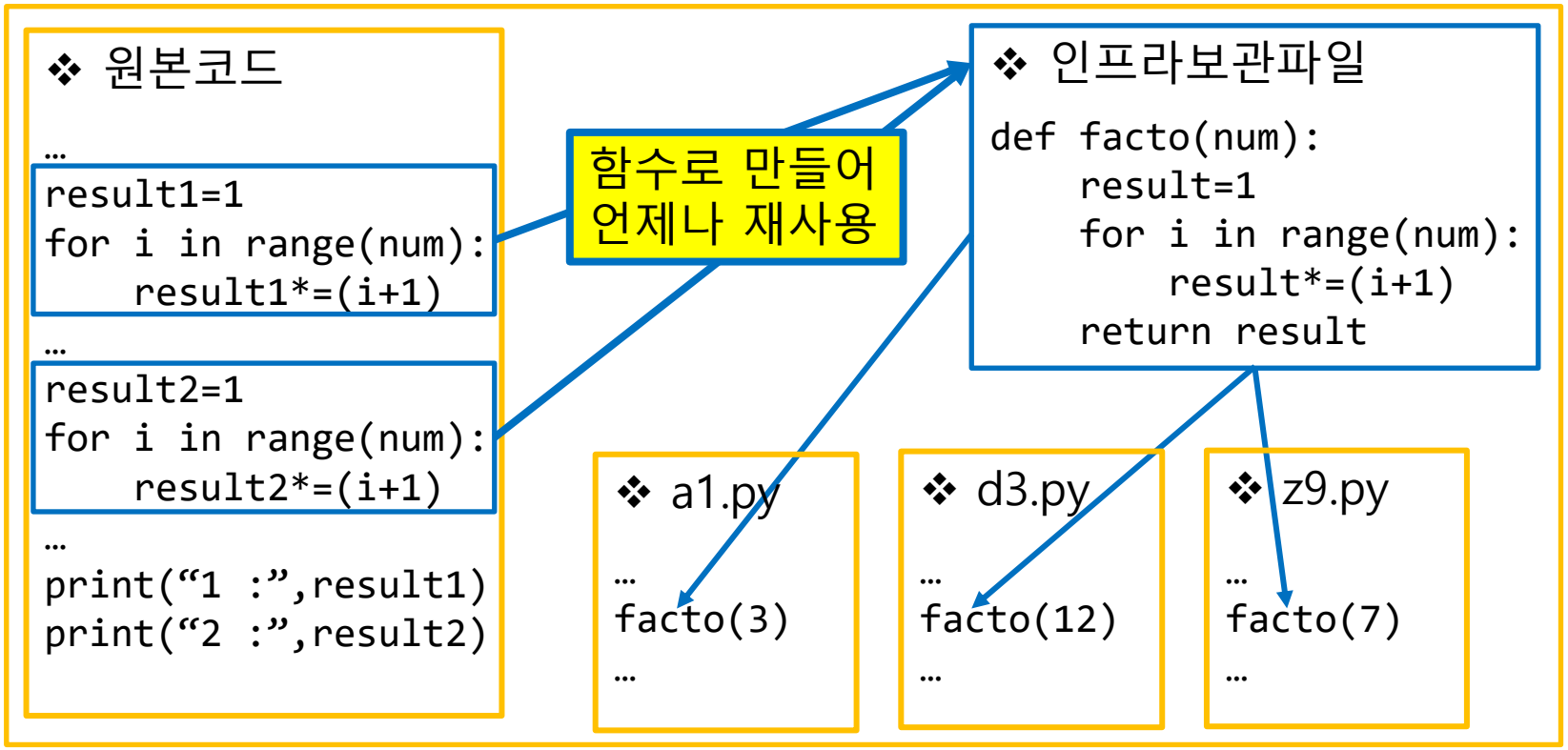
함수

함수(Function)

❖ **하나의 작업을 수행하도록 구성한 코드의 집합**

- **불규칙적으로 반복되는 같은 코드**를 줄이기 위한 방법
- 특정 작업을 하는 코드의 집합에 **이름을 붙여 사용하는 것**
- **나만의 인프라를 구축하여 작업효율을 올리기 위한 것**

❖ **개념도**



함수(Function)

❖ 함수의 필수조건

- **하나의 기능**으로 구성 : 다양한 기능을 수행하지 않음.
- **독립적**인 코드 : 다른 코드에 종속되지 않음
- **재사용**이 가능 : 한번만 쓰이는 코드는 의미 없음

❖ 함수의 특징

- 다른 코드와 **격리된 공간**을 보유 : 공간을 **공유하지 않음**
- 격리된 공간을 오가기 위한 **출구와 입구**가 존재

❖ 함수의 장점

- **기능단위로 구분되어 코드의 가독성이 상승**
- 에러 및 버그의 관리 및 해결이 쉬워짐

함수(Function)

❖ **함수의 생성 : 함수를 정의하다**

- 함수 이름, 처리할 값, 처리과정 등
- 일련의 코드를 작성하여 준비

❖ 코드

```
def facto(num):
    result=1
    for i in range(num):
        result*=(i+1)
    return result
```

❖ **함수 정의시 구성요소**

① **def** : 작성하는 코드가 함수임을 알리기 위한 예약어

② **함수명** : 해당 기능에 이름을 붙이는 것. **동사** 사용을 권장

③ **매개변수** : 어떤 값을 받아 처리할 수 있도록 준비된 변수

- 파이썬에서 **변수만 준비하는 것이 허용되는 곳. 생략 가능**

④ **종속문장** : 함수가 **수행할 기능을 작성한 코드**를 의미

- **들여쓰기**를 통하여 함수에 종속되었음을 식별

⑤ **return** : 해당 기능이 수행된 후에 돌려줄 **결과물 지정**

- 필요하다면 있어야 하고, 필요가 없다면 생략이 가능

함수(Function)

❖ 함수의 사용 : 함수를 호출하다

- 격리된 공간에 있는 함수를 불러옴
- 함수는 해당 지점에서 기능을 수행

❖ 코드

```
num=5
result=facto(num)
print(result)
```

❖ 함수의 호출시 구성요소

① **전달인자** : 처리하기 위해 넘기는 **값**

- ➢ 매개변수의 유무에 따라 사용여부가 결정됨

② **함수명** : 어떤 함수를 호출할 것인지 **이름**으로 지정

- ➢ 함수를 불러오는데 메모리주소는 비효율적

③ **result** : 함수가 처리한 **결과**를 사용하기 위한 임의 변수

- ➢ return의 유무에 따라 사용여부가 결정됨

함수(Function)

❖ **함수에는 어느정도 준수하는 형식이 존재**

- **반드시 지켜야 하는 것은 아니며 변동될 수 있음**

① **매개변수 있음, return 있음**

❖ 값의 연산 혹은 처리 **과정**을 함수로 만들 때 이용

❖ 연산이나 처리는 **연결되는 과정**이기 때문에 둘 다 필요

② **매개변수 있음, return 없음**

❖ 값의 연산 혹은 처리 **결과**를 함수로 만들 때 이용

❖ return이 없어서 반드시 **자체적인 출력**이 필요함

③ **매개변수 없음, return 있음**

❖ 값의 연산 혹은 처리 **시작**을 함수로 만들 때 이용

❖ 매개변수가 없어 **자체적으로 값을 준비하거나 입력**을 받음

④ **매개변수 없음, return 없음**

❖ 지나치게 긴 코드를 줄이기 위한 용도.

함수(Function)

< 파일이름 : functionEX1.py >

❖ 실습예제1. 아래의 조건을 달성하는 코드를 작성하세요.

조건

1. 두 수의 합을 구하는 함수를 정의하세요.
2. 두 수의 차를 구하는 함수를 정의하세요.
3. 두 수의 곱을 구하는 함수를 정의하세요.
4. 두 수의 몫을 구하는 함수를 정의하세요.

*위의 함수들은 모두 매개변수와 return이 있습니다.

결과

```
--출력--
_sum((값1),(값2)) : (결과1) # 합
_sub((값1),(값2)) : (결과2) # 차
_mul((값1),(값2)) : (결과3) # 곱
_div((값1),(값2)) : (결과4) # 몫
```

함수(Function)

< 파일이름 : functionEX2.py >

❖ 실습예제2. 아래의 조건을 달성하는 코드를 작성하세요.

조건

1. 두 수의 비교해서 더 큰 수를 출력하는 함수를 정의하세요.
2. 하나의 정수에 대하여 절대값을 출력하는 함수를 정의하세요.
3. 두 수의 몫, 나머지, 나누기를 출력하는 함수를 정의하세요.

*위의 함수들은 모두 매개변수는 있습니다.
*위의 함수들은 모두 return이 없습니다.

결과(값1 : 6, 값2 : 3, 값3 : -8로 가정)

```
--출력--
(값1)이 더 큽니다.                    # _comp((값1),(값2))
(값3)의 절대값은 (절대값3) 입니다. # _abs_num((값3))
결과 : (몫), (나머지), (나누기)    # _div3((값1), (값2))
```

함수(Function)

< 파일이름 : functionEX3.py >

❖ **실습예제3. 아래의 조건을 달성하는 코드를 작성하세요.**

조건

1. 값을 3번 입력을 받아 리스트를 만드는
   함수를 정의하세요.
2. 키, 값을 2번 입력을 받아 딕셔너리를 만드는
   함수를 정의하세요.

*위의 함수들은 모두 매개변수가 없습니다.
*위의 함수들은 모두 return이 있습니다.

결과

--출력--

make_lst() 사용시
결과 : [(값1), (값2), (값3)]
make_dic() 사용시
결과 : {(키1):(값1), (키2):(값2)}

함수(Function)

< 파일이름 : functionEX4.py >

❖ **실습예제4. 아래의 조건을 달성하는 코드를 작성하세요.**

조건(*직접 임의로 구성합니다.)

1. 리스트 내부에 있는 정수문자열을 정수로 바꾸는 함수를 정의하세요.
2. 리스트 내부에 있는 실수문자열을 실수로 바꾸는 함수를 정의하세요.
3. 리스트 내부에 있는 값의 합을 구하는 함수를 정의하세요.
4. 리스트 내부에 있는 값의 평균을 구하는 함수를 정의하세요.

결과(['1','2','3','4','5'], ['1.1','2.2','3.3','4.4','5.5'])

--출력--

```
정수리스트              : [1, 2, 3, 4, 5]
정수리스트의 합         : 15
실수리스트              : [1.1, 2.2, 3.3, 4.4, 5.5]
실수리스트의 평균       : 3.3
```