인공지능 기초 프로그래밍

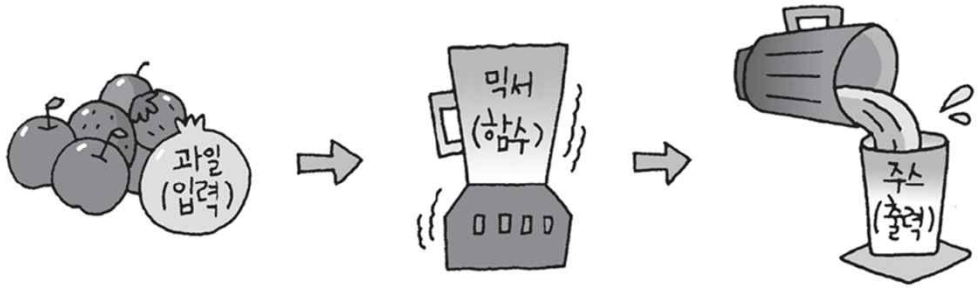
Python 기초 (함수)

파이썬 프로그램 기초-함수

▪ 함수란?

- 우리는 믹서에 과일을 넣고, 믹서를 사용해서 과일을 갈아 과일 주스를 만듦

- 믹서에 넣는 과일 = 입력
- 과일주스 = 출력
- 믹서 = ?



믹서는 과일을 입력받아 주스를 출력하는 함수와 같다.

파이썬 프로그램 기초-함수

▪ 함수란?

- 함수가 하는 일

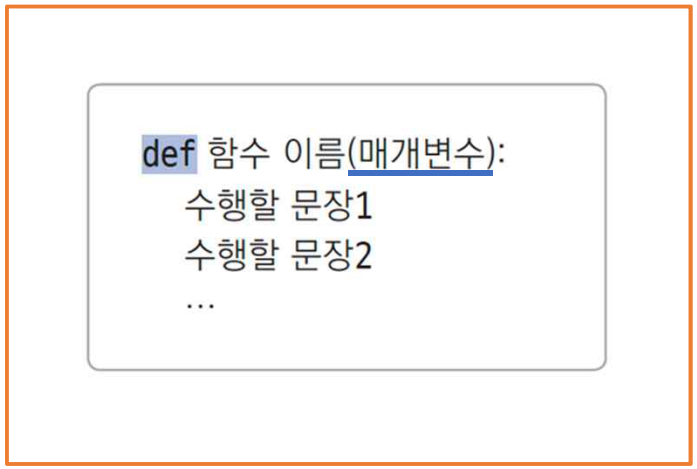
> 입력값을 가지고 어떤 일을 수행한 다음에 그 결과물을 내어놓는 것

- 함수를 사용하는 이유
  - 반복되는 부분이 있을 경우 '반복적으로 사용되는 가치 있는 부분'을 한 뭉치로 묶어서 '어떤 입력값을 주었을 때 어떤 결괏값을 돌려준다'라는 식의 함수로 작성하는 것이 현명함
  - 프로그램의 흐름을 파악하기 좋고 오류 발생 지점도 찾기 쉬움

▪ 파이썬 함수의 구조

- def : 함수를 만들 때 사용하는 예약어
- 함수 이름은 임의로 생성 가능
- 매개변수는 함수에 입력으로 전달되는 값을 받는 변수

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



- return : 함수의 결괏값을 돌려주는 명령어

함수의 이름은 "add"이며, "두 개의 파라미터"를 입력 받아,
입력 받은 파라미터들을 "더한 결과 값을 반환"한다.

파이썬 프로그램 기초-함수

▪ 파이썬 함수의 구조

- 예) add 함수
  - add 함수 만들기

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

  - add 함수 사용하기

```
>>> a = 3
>>> b = 4
>>> c = add(a, b)  ◀── add(3, 4)의 반환 값을 c에 대입
>>> print(c)
7
```

▪ 매개변수와 인수

- **매개변수**와 **인수**는 혼용해서 사용되는 헷갈리는 용어로 잘 구분하는 것이 중요!

- **매개변수(parameter)**
  - 함수에 입력으로 전달된 값을 받는 변수

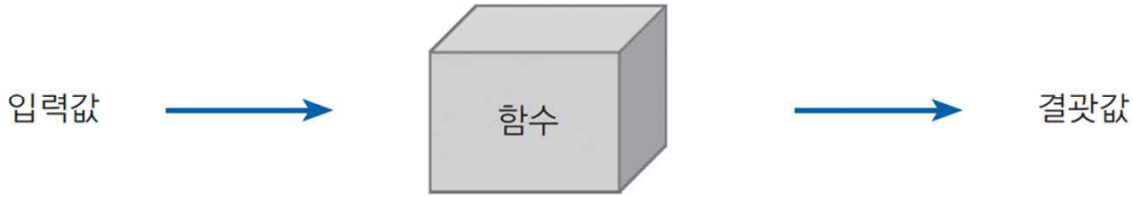
- **인수(arguments)**
  - 함수를 호출할 때 전달받는 입력값

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

print(add(3, 4)) ◀── 3, 4는 인수
```


▪ 입력값과 결괏값에 따른 함수 형태

- 함수는 들어온 입력값을 받아 어떤 처리를 하여 적절한 결과값을 돌려줌



- 함수의 형태는 입력값과 결과값의 존재 유무에 따라 4가지 유형으로 나뉨

	입력	출력
(1)	O	O
(2)	X	O
(3)	O	X
(4)	X	X

▪ 입력값과 결괏값에 따른 함수 형태

1. 일반적인 함수 (1)

- 입력값이 있고 결괏값이 있는 함수

```
def 함수 이름(매개변수):
    수행할 문장
    …
    return 결괏값
```

- 일반 함수의 전형적인 예
  - add 함수는 2개의 입력값을 받아서 서로 더한 결괏값을 돌려줌

```
>>> def add(a, b):
...     result = a + b
...     return result  ◀── a+b의 결괏값 반환
...
>>>
```

```
>>> a = add(3, 4)
>>> print(a)
7
```

결괏값을 받을 변수 = 함수이름(입력인수 1, 입력인수 2, …)

▪ 입력값과 결괏값에 따른 함수 형태

2. 입력값이 없는 함수 (2)

- 입력값이 없는 함수도 존재함

- say 함수는 매개변수 부분을 나타내는 함수 이름 뒤의 괄호 안이 비어있음
- 함수 사용 시 say( )처럼 괄호 안에 아무 값도 넣지 않아야 함

```
>>> def say():
...     return 'Hi'
...
>>>
```

```
>>> a = say()
>>> print(a)
Hi
```

결괏값을 받을 변수 = 함수이름()

▪ 입력값과 결괏값에 따른 함수 형태

3. 결과값이 없는 함수 (3)

- 결과값이 없는 함수는 호출해도 돌려주는 값이 없음

```
>>> def add(a, b):
...     print("%d, %d의 합은 %d입니다." % (a, b, a+b))
...
>>>
```

```
>>> add(3, 4)
3, 4의 합은 7입니다.
```

return 값을 a 변수에 대입하여 출력하여 확인

None이란 거짓을 나타내는 자료형으로, 결과값이 없을 때 쓰이는 반환 값

```
>>> a = add(3, 4)
3, 4의 합은 7입니다.
>>> print(a)
None
```

▪ 입력값과 결괏값에 따른 함수 형태

4. 입력값도 결과값도 없는 함수 (4)

- 입력 인수를 받는 매개변수도 없고 return문도 없는, 즉 입력값도 결괏값도 없는 함수

```
>>> def say():
...     print('Hi')
...
>>>
```

```
>>> say()
Hi
```

함수이름()

▪ 매개변수 활용법

- 함수 호출 시 매개변수 지정 가능
  - 예) add 함수

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

- 매개변수 지정하여 사용

```
>>> result = add(a=3, b=7)  ◀── a에 3, b에 7을 전달
>>> print(result)
10
```

- 매개변수를 지정하면 매개변수 순서에 상관없이 사용할 수 있다는 장점이 있음

```
>>> result = add(b=5, a=3)  ◀── b에 5, a에 3을 전달
>>> print(result)
8
```

파이썬 프로그램 기초-함수

- 입력 값이 몇 개가 될지 모를 때

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

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

rtn_val = []
rtn_val.append(add_many(1,2,3,4,5))
rtn_val.append(add_many(1,2,3,4))
rtn_val.append(add_many(1,2,3))
print(rtn_val)
```

```
val = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
tmp = add_many(*val)
print(tmp)
```

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

print(add_mul('add', 1, 2, 3, 4, 5))
print(add_mul('mul', 1, 2, 3, 4, 5))
```

사칙연산 operation과 입력 값들을 받아 출력하는 함수 작성

파이썬 프로그램 기초-함수

- 키워드 파라미터
  - 함수의 인자로 " 변수=값" 형태로 받고 싶을 경우

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

```
def func_kwargs(**kwargs):
  print(kwargs)

func_kwargs(a=1, b=3)


def myFunc(**kwargs):
  for item in kwargs.items():
    print(item)

myFunc(x=100, y=200, z='abc')
```

파이썬 프로그램 기초-함수

- 함수의 결과 값은 언제나 하나

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

print("result=%d,%d" % add_and_mul(2, 4))
result = add_and_mul(2, 4)
a, b = add_and_mul(2, 4)
print(result, a, b)
```

```
result=6,8
(6, 8) 6 8
```

튜플 값으로 반환

교재 160p
Return statement 2개: 당연히 첫번째 데이터 반환 후 함수 종료

▪ 초깃값 설정하기

- 매개변수에 초깃값을 미리 설정

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

say_myself 함수는 3개의 매개변수를 받아서 마지막 인수인 man이 True이면 "남자입니다.", False이면 "여자입니다."를 출력한다.

- man=True : 매개변수에 미리 값을 넣어줌 → 함수의 매개변수 초깃값을 설정하는 방법

▪ 초깃값 설정하기

- 매개변수에 들어갈 값이 항상 변하는 것이 아니면, 초깃값을 미리 설정하는 것이 유용함
- say_myself 함수 사용법

```
say_myself("박응용", 27)
say_myself("박응용", 27, True)
```

```
나의 이름은 박응용입니다.
나이는 27살입니다.
남자입니다.
```

- 입력값으로 "박응용", 27처럼 2개를 주면, name에는 "박응용"이 old에는 27이 대입됨
- man이라는 변수에는 입력값을 주지 않았지만 초깃값 True를 갖게 됨

주의사항: 매개변수 초기 값 설정 시
"초기 값을 설정해 놓은 매개변수 뒤에 초기 값을 설정해 놓지 않은 매개 변수는 사용할 수 없음"
e.g.) def say_myself(name, man=True, old)

▪ 함수 안에서 선언한 변수의 효력 범위

- 함수 안에서 사용할 변수의 이름을 함수 밖에서도 동일하게 사용한다면?

```
# vartest.py
a = 1 ◀── 함수 밖의 변수 a
def vartest(a):
    a = a +1
```

vartest 함수 선언

```
vartest(a) ◀── vartest 함수의 입력값으로 a를 줌
print(a) ◀── a 값 출력
```

- 함수를 실행해 보면, 결괏값은 **1**이 나옴
- 함수 안에서 새로 만든 매개변수는 함수 안에서만 사용하는 **'함수만의 변수'**이기 때문
- 즉, 매개변수 a는 함수 안에서만 사용하는 변수로, 함수 밖의 변수 a가 아님

파이썬 프로그램 기초-함수

- **lambda**
  - 함수를 생성할 때 사용하는 예약어
  - def와 동일한 역할
  - '람다'

```
lambda 매개변수1, 매개변수2, ... : 매개변수를 사용한 표현식
```

  - 보통 함수를 한 줄로 간결하게 만들 때 사용
  - def를 사용해야 할 정도로 복잡하지 않거나 def를 사용할 수 없는 곳에 주로 쓰임

파이썬 프로그램 기초-함수

- **lambda**
  - 사용 예시

```
>>> add = lambda a, b: a+b
>>> result = add(3, 4)
>>> print(result)
7
```

    - add는 두 개의 인수를 받아 서로 더한 값을 돌려주는 lambda 함수

  - def를 사용한 경우와 하는 일이 완전히 동일함

```
>>> def add(a, b):
...     return a+b
...
>>> result = add(3, 4)
>>> print(result)
7
```

파이썬 프로그램 기초-함수

교재 168~: 04-2 사용자 입출과 출력, 파일 읽고 쓰기.

```
>>> a = input()
Life is too short, you need python
>>> a
'Life is too short, you need python'
>>>
```

input()은 입력되는 모든 것을 문자열로 취급

사용자가 입력한 문장을 a에 대입

파일 객체 = open(파일 이름, 파일 열기 모드)

파일 열기 모드	설명
r	읽기 모드 - 파일을 읽기만 할 때 사용
w	쓰기 모드 - 파일에 내용을 쓸 때 사용
a	추가 모드 - 파일의 마지막에 새로운 내용을 추가할 때 사용

- readline문: 라인 읽기
- read문: 파일 전체 읽기
- with문: f.open, f.close 자동

Practice #1 ()

Function_example

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

rtn_val = []
rtn_val.append(add_many(1,2,3,4,5))
rtn_val.append(add_many(1,2,3,4))
rtn_val.append(add_many(1,2,3))
print(rtn_val)

val = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
tmp = add_many(*val)
print(tmp)
```

함수 범위

```
# 함수의 범위
a = 1
def vartest(a):
  a = a + 1

vartest(a)
print(a)

def vartest1(inner_var):
  inner_var = inner_var + 1

vartest1(3)
# print(inner_var)
```

Global / lambda

```
# global
aa = 1
def globaltest(a):
  global aa
  aa = a + 1

globaltest(3)
print(aa)

#lambda
add = lambda a, b: a + b
result = add(3, 4)
print(result)
```

Thank you

Q&A

www.kopo.ac.kr
jsshin7@kopo.ac.kr

