함수란 ¶

- 함수란 입력변수와 출력변수간의 대응 관계를 정의한 것을 말한다.
- 프로그램에서 함수란 하나의 작업, 기능, 동작을 처리하기 위한 사용자 정의 연산자라고 할 수 있다.
  - 함수는 값을 입력(Input)을 받아서 처리후 처리결과를 출력(Output)하는 일련의 과정을 정의한 것을 말한다.
  - 만들어진 함수는 동일한 작업이 필요할 때 마다 재사용될 수 있다.
  - 함수를 구현해 파이썬 실행환경에 등록하는 것을 **함수를 정의(define)한다** 라고 한다.
  - 정의된 함수를 사용하는 것을 **함수를 호출(call)한다** 라고 한다.
  - 파이썬에서 함수는 일급 시민(First Class Citizen)이다.

> - **일급 시민 이란**
> – 변수에 할당할 수 있고, 함수의 입력값으로 전달할 수 있고, 함수의 반환 값으로 반환할 수 있는 객체를 말한다.

함수 만들기

- 함수의 정의
  - 새로운 함수를 만드는 것을 함수의 정의라고 한다.
  - 함수를 구현하고 그것을 파이썬 실행환경에 새로운 기능으로 등록하는 과정을 말한다.

- 함수 구현
  - 함수의 선언부와 구현부로 나누어진다
    - 함수의 선언부(Header) : 함수의 이름과 입력값을 받을 변수(Parameter, 매개변수)를 지정한다.
    - 함수의 구현부(Body) : 함수가 호출 되었을 때 실행할 실행문들을 순서대로 작성한다.

```
def 함수이름( [변수, 변수, ..]):  # 선언 부(Header)
    # 구현 부(body)
    실행구문1
    실행구문2
    실행구문3
    ...
    [return [결과값]]
```

- 함수 선언 마지막에는 : 을 넣어 구현부와 구분한다.
- Parameter(매개변수)는 argument(호출하는 곳에서 전달하는 함수의 입력값)를 받기 위한 변수로 0개 이상 선언할 수 있다.
- 함수의 실행구문은 코드블록으로 들여쓰기로 블록을 묶어준다.
  - 들여쓰기는 보통 공백 4칸을 사용한다.
- 함수의 처리 결과값이 있을 경우 **return 구문**을 넣고 없을 경우 return은 생략할 수 있다.
- **함수이름 관례**
  - 함수이름은 보통 동사형으로 만든다.
  - 모두 소문자로 하고 여러단어로 구성할 경우 각 단어들을 _ 로 연결한다. (변수와 동일)

In [ ]:

In [ ]:

return value(반환값)

- 함수가 호출받아 처리한 결과값으로 호출한 곳으로 반환하는 값이다.
- 함수 구현부에 return [값] 구문을 사용해 반환한다.
  - **return**
    - 함수가 정상적으로 끝났고 호출한곳으로 돌아간다.
    - 보통은 함수 구현의 마지막에 넣지만 경우에 따라 중간에 올 수 있다.
  - return 반환값
    - 호출한 곳으로 값을 가지고 돌아간다. (반환한다)
    - 반환값이 없을 경우 None을 반환한다.
    - 함수에 return 구문이 없을 경우 마지막에 return None이 실행된다.
- 여러개의 값을 return 하는 경우 자료구조로 묶어서 전달해야한다.
  - 함수는 한개의 값만 반환할 수 있다.

In [ ]:

In [ ]:

Parameter (매개변수)

기본값이 있는 Parameter

- 매개변수에 값을 대입하는 구문을 작성하면 호출할 때 argument 가 넘어오지 않으면 대입해놓은 기본값을 사용한다.
- 함수 정의시 기본값 없는 매개변수, 있는 매개변수를 같이 선언할 수 있다.
  - **이때 기본값 없는 매개변수들을 선언하고 그 다음에 기본값 있는 매개변수들을 선언한다.**

In [ ]:

Positional argument와 Keyword argument

- Argument는 함수/메소드를 호출할 때 전달하는 입력값을 말한다.
  - Argument는 전달하는 값이고 Parameter는 그 값을 저장하는 변수
- Positional argument

    - 함수 호출 할때 argument(전달인자)를 Parameter 순서에 맞춰 값을 넣어서 호출.
- keyword argument
    - 함수 호출할 때 argument를 `Parameter변수명 = 전달할값` 형식으로 선언해서 어떤 parameter에 어떤 값을 전달할 것인지 지정해서 호출.
    - 순서와 상관없이 호출하는 것이 가능.
    - parameter가 많고 대부분 기본값이 있는 함수 호출 할 때 뒤 쪽에 선언된 parameter에만 값을 전달하고 싶을 경우 유용하다.

In [ ]:

```
```

가변인자 (Var args) 파라미터

- 호출하는 쪽에서 argument로 0 ~ n개의 값을 나열해서 여러개의 값들을 전달하면 **tuple이나 dictionary로 묶어서** 받을 수있도록 선언하는 parameter
    - positial argument로 전달하는 것과 keyword argument로 전달되는 값을 받는 두가지 방식이 있다.
- *변수명: **positional argument**를 개수와 상관없이 하나의 변수로 받을 수 있도록 선언하는 가변인자.
    - 전달된 값들은 tuple로 받아서 처리한다.
    - 관례적으로 변수명은 *args 를 사용한다.
- **변수명: **keyword argument**를 개수와 상관없이 하나의 변수로 받을 수 있도록 선언하는 가변인자.
    - 전달된 값들은 dictionary로 받아서 처리한다.
    - 관례적으로 변수명은 **kwargs 를 사용한다.
- 하나의 함수에 가변인자는 * 하나, ** 두개짜리 각각 한개씩만 선언할 수 있다.
- 파라미터 선언순서
    1. 기본값이 없는 파라미터
    2. 기본값이 있는 파라미터
    3. `*args`
    4. `**args`

In [ ]:

```
```

변수의 유효범위

- **지역변수 (local variable)**
    - 함수안에 선언된 변수
    - 선언된 그 함수 안에서만 사용할 수 있다.
- **전역변수 (global variable)**
    - 함수 밖에 선언 된 변수
    - 모든 함수들이 공통적으로 사용할 수 있다.
    - 하나의 함수에서 값을 변경하면 그 변한 값이 모든 함수에 영향을 주기 때문에 함부로 변경하지 않는다.
    - 함수내에서 전역변수에 값을 대입하기 위해서는 global 키워드를 이용해 사용할 것을 미리 선언해야 한다.
        - global로 선언하지 않고 함수안에서 전역변수와 이름이 같은 변수에 값을 대입하면 그 변수와 동일한 지역변수을 생성한다.
        - 조회할 경우에는 상관없다.

함수에서 변수를 조회할 경우 **먼저 지역변수를 찾고 없으면 전역변수를 찾는다.**

In [ ]:

함수는 일급시민(First class citizen) 이다.

- 일급 시민
  1. 변수에 대입 할 수 있다.
  2. Argument로 사용할 수 있다.
  3. 함수나 메소드의 반환값으로 사용 할 수 있다.
- 일급
  - 모든 권리를 다 가진다는 의미
- 시민
  - 프로그래밍 언어를 구성하는 객체를 의미
- 즉 파이썬에서 함수는 일반 값(객체) 로 취급된다.

In [ ]:

람다식/람다표현식 (Lambda Expression)

- 함수를 하나의 식을 이용해서 정의할때 사용하는 표현식(구문).
- 값을 입력받아서 **간단한 처리한 결과**를 반환하는 간단한 함수를 표현식으로 정의할 수 있다.
  - 처리결과를 return 하는 구문을 하나의 명령문으로 처리할 수 있을때 람다식을 사용할 수 있다. => return x + y
- 구문

```
lambda 매개변수[, 매개변수, ...] : 명령문(구문)
```

- 명령문(구문)은 하나의 실행문만 가능하다.
- 명령문(구문)이 처리한 결과를 리턴해준다.
- **람다식은 함수의 매개변수로 함수를 전달하는 일회성 함수를 만들때 주로 사용한다.**

In [ ]:

iterable 관련 함수에서 함수를 매개변수로 받아 처리하는 함수들

- sorted(iterable, reverse=False, key=None)
  - 정렬처리
  - 매개변수
    - reverse: True - 내림차순, False - 오름차순(기본)
    - key: 함수
      - Parameter로 iterable의 각 원소를 받는 함수.
      - 정렬을 할 때 iterable의 원소기준으로 정렬하는 것이 아니라 이 함수가 반환하는 값을 기준으로 정렬

- filter(함수, Iterable)
  - Iterable의 원소들 중에서 특정 조건을 만족하는 원소들만 걸러주는 함수
  - 함수
    - 어떻게 걸러낼 것인지 조건을 정의. 매개변수 1개, 반환값 bool
    - 원소 하나 하나를 함수에 전달해 True를 반환하는 것만 반환
- map(함수, Iterable)
  - Iterable의 원소들 하나 하나를 처리(변형)해서 그 결과를 반환
  - 함수
    - 원소들을 어떻게 처리할지 정의. 매개변수 1개. 반환값: 처리 결과
- **filter/map 반환타입**: generator 형태로 반환 된다. (리스트가 아님)

In [ ]:

docstring

- 함수에 대한 설명
- 함수의 구현부의 첫번째에 여러줄 문자열로 작성한다.
- 함수 매개변수/리턴타입에 대한 힌트(주석)

In [ ]:

pass 키워드(예약어)

- 빈 구현부를 만들때 사용
  - 코드블럭을 하는 일 없이 채울 때 사용
  - `...` 을 대신 사용할 수 있다.

TODO

In [ ]:

```
# 1. 사용자가 입력한 단의 구구단을 출력하는 함수를 구현(매개변수로 단을 받는다.)


# 2. 시작 정수, 끝 정수를 받아 그 사이의 모든 정수의 합을 구해서 반환하는 함수를 구현(ex: 1, 20 =


# 3. 1번 문제에서 시작을 받지 않은 경우 0을, 끝 정수를 받지 않으면 10이 들어가도록 구현을 변경


# 4. 체질량 지수는 비만도를 나타내는 지수로 키가 a미터 이고 몸무게가 b kg일때 b/(a**2) 로 구한다.
# 체질량 지수가
#- 18.5 미만이면 저체중
#- 18.5이상 25미만이면 정상
#- 25이상이면 과체중
#- 30이상이면 비만으로 하는데
#몸무게와 키를 매개변수로 받아 비만인지 과체중인지 반환하는 함수를 구현하시오.



# 람다식
#5. filter()를 이용해 다음 리스트에서 양수만 추출히 리스트를 구현
ex1 = [1, -10, -2, 20, 3, -5, -7, 21]


#6. filter()와 map()을 이용해 다음 리스트에서 음수만 추출한 뒤 그 2 제곱한 값들을 가지는 리스트를
ex2 = [1, -10, -2, 20, 3, -5, -7, 21]
```