21장

함수로 모듈화와 추상화하기

21장 함수로 모듈화와 추상화하기

21.1 함수 작성하기

21.1 함수 작성하기

함수를 작성할 때는 세 가지를 생각해야 한다.

- 함수가 받을 입력은 무엇인가?
- 함수가 어떤 계산/연산을 수행할 것인가?
- 함수가 어떤 결과를 돌려줄 것인가?

21.1 함수 작성하기

출석 부르기 예제를 생각해 보자.

- 함수는 def라는 키워드로 시작한다. 함수 본문의 첫 부분은 보통 3중 따옴표(여러줄문자열)를 사용해 함수가 하는 일을 문서화하는 부분으로 이뤄진다. 3중 따옴표 안에는 함수 입력이 무엇인지, 함수가 하는 일이 무엇인지, 함수가 반환하는 값이 무엇인지 등 함수가 하는 일에 대한 설명을 적는다.
- 한 반에 속한 모든 학생이 물리적으로 교실 안에 있는지 for 루프를 사용해 검사한다. 루프 안에서 학생이 물리적으로 교실 안에 있다는 조건을 만족하는 경우, 그 학생의 이름을 출력한다. for 루프가 끝나고 나면 출석 부르기 끝이라는 문자열을 반환한다.

```
def take_attendance(classroom_roaster, who_is_here): ---- 함수 정의
    """
    classroom_roaster, tuple
    who_is_here, tuple

    classroom_roaster(출석부)에 있는 모든 사람이
    who_is_here(현재 교실에 있는 사람 목록)에 들어 있는지 검사한다
    학생이 who_is_here에 있는 경우에만 그 이름을 출력한다

    Returns "출석 부르기 끝"
    """
    for kid in classroom_roaster: ---- 출석부에 있는 모든 학생에 대해 루프를 반복
        if kid in who_is_here: ---- kid가 who_is_here 튜플 안에 들어 있는지 검사
            print(kid) ---- 교실에 있는 학생 이름을 출력
    return "출석 부르기 끝" ---- 문자열을 반환
```

함수 명세(독스트링)

코드 21-1.
출석을 부르는 함수

21.1 함수 작성하기

셀프 체크 21.1

21.1.1 함수 기초: 함수 입력

- 함수의 모든 입력은 매개변수(parameter)라는 이름의 변수다. 이를 인자(argument)라고도 부른다. 또 형식 인자(formal arguments)라고 더 구체적으로 부르기도 한다.

- 형식이라는 단어를 붙이는 이유는 함수 정의 안에서 이들이 구체적인 값을 가지지 않기 때문이다.

- 매개변수에 실제로 값이 대입되는 때는 여러분이 함수를 호출하면서 값을 넘길 때다.

- 셀프 체크 21.2

21.1.2 함수 기초: 함수가 하는 일

- 함수를 작성할 때는 모든 함수 매개변수에 값이 들어가 있다고 가정하고 함수 본문의 코드를 작성한다.
- 함수 구현은 들여 써야 한다는 점을 제외하면 그냥 파이썬 코드에 불과하다.
- 프로그래머는 원하는 대로 함수를 구현할 수 있

- 셀프 체크 21.3

21.1.3 함수 기초: 함수가 반환하는 값

함수는 일을 해야 한다.

- 함수를 사용하면 같은 동작을 입력을 바꿔가면서 반복 수행할 수 있다.
- 따라서 함수 이름은 get_something, set_something, do_something처럼 일반적으로 그 함수가 하는 일을 설명하는 동사나 동사구로 만드는 경우가 많다.

함수는 자신만의 환경을 만든다.

- 그 환경에서 만들어진 변수는 함수 바깥에서는 읽거나 쓸 수 없다.
- 함수를 사용하는 목적은 어떤 작업을 수행하고 결과를 돌려주는 것이다. 파이썬에서는 return키워드(return은 반환한다는 뜻의 동사 단어다)를 사용해 결과를 돌려준다.
- return 키워드가 들어있는 문장은 그 문장이 들어 있는 함수가 처리를 마쳤고, 프로그램에서 함수를 불렀던 쪽에 결과를 돌려줄 준비가 되었다는 뜻이다.

```
def get_word_length(word1, word2): ---- 매개변수를 두 개 받는 함수 정의
    word = word1+word2 ---- 두 매개변수를 이어 붙임
    return len(word) ---- return 문: 두 문자열을 연결한 문자열 길이 반환
    print("이 문자열은 절대로 출력되지 않습니다") ---- return 문 뒤에 있는 문장은 실행되지 않음
```

코드 21-2
두 문자열을 연결한 문자열의 길이를 알려주는 함수

21.1.3 함수 기초: 함수가 반환하는 값

» 셀프 체크 21.5

21.2 함수 사용하기

21.2 함수 사용하기

함수를 호출하는 방법

- 함수이름을 먼저 쓰고, 그 뒤에 괄호를 넣고, 괄호 사이에 인자(argument)또는 실인자(actual parameter) 를 넣으면 된다.
- 인수로는 값을 직접 넣거나, 값이 지정된 변수나 값을 계산할 수 있는 식을 넣으면 된다.

```
def word_length(word1, word2):
    word = word1+word2
    return len(word)
    print("이 문자열은 절대로 출력되지 않습니다")

length1 = word_length("Rob", "Banks")
length2 = word_length("Barbie", "Kenn")
length3 = word_length("Holly", "Jolley")

print("One name is", length1, "letters long.")
print("Another name is", length2, "letters long.")
print("The final name is", length3, "letters long.")
```

word_length 함수 정의

각 줄은 여러 다른 입력으로 함수를 호출하고,
함수가 반환한 결과를 변수에 저장함

각 줄은 변수를 출력

코드 21-3
함수를 호출하는 방법

21.2 함수 사용하기

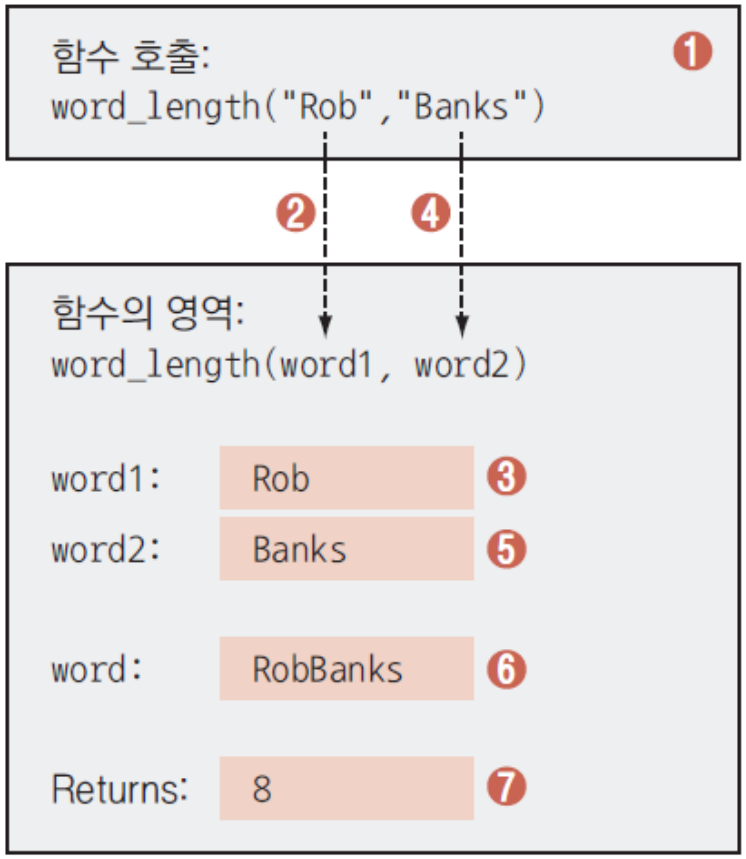

그림 21-1
❶에서 함수를 호출하면 어떤 일이 생길까?
❷와 ❸을 보면 첫 번째 실인자가 함수 영역 안에서 첫 번째 형식 인자에 연관됨을 알 수 있다.
마찬가지로 ❹ 와 ❺에서 두 번째 실인자와 함수의 두 번째 형식 인자가 연관된다. ❻은 함수 내부에서 만들어진 새 변수다.
❼은 값을 반환하는 문장이다. 함수가 반환되면 함수의 영역이 없어진다. 그때 함수 안에 있던 변수들도 사라진다

21.2.1 둘 이상의 값 반환하기

튜플을 사용하면 함수가 둘 이상의 값을 반환하도록 '트릭'을 걸 수 있다.

- 튜플의 각 원소에 다른 값을 넣으면 된다.
- 이 방법을 쓰면 함수는 객체 하나(튜플)만 반환하지만, 튜플 안에 원하는 대로 여러 다른 값들(원소)을 넣을 수 있다.

함수를 만든 뒤 함수를 호출해서 반환 받은 튜플의 각 원소를 코드 21-4처럼 여러 변수에 저장할 수 있다.

- add_sub 함수는 두 매개변수를 더하고 뺀 다음, 튜플에 두 연산의 결과를 저장해 반환한다.
- add_sub 함수를 호출한 후 결과를 (a,b)라는 다른 튜플에 저장하면 a에 덧셈 결과, b에 뺄셈 결과가 담긴다

```
def add_sub(n1, n2):
    add = n1 + n2
    sub = n1 - n2
    return (add, sub) ---- 덧셈과 뺄셈 결과가 담긴 튜플을 반환
(a, b) = add_sub(3,4) ---- 결과를 튜플에 저장
```

코드 21-4 튜플 반환하기

21.2.1 둘 이상의 값 반환하기

셀프 체크 21.6

```
def guessed_card(number, suit, bet):
    money_won = 0
    guessed = False
    if number == 8 and suit == "hearts":
        money_won = 10*bet
        guessed = True
    else:
        money_won = bet/10
    # 두값을 반환하는 문장을 작성하라.
    # (1) 이긴 경우 상금액
    # (2) 지정한 number와 suit이 맞았는지 여부

print(guessed_card(8, "hearts", 10))
print(guessed_card("8", "hearts", 10))
guessed_card(10, "spades", 5)
(amount, did_win) = guessed_card("eight", "hearts", 80)
print(did_win)
print(amount)
```

21.2.2 return 문이 없는 함수

파이썬에서는 함수 본문의 return 문을 생략해도 된다.

- return 문을 쓰지 않으면 파이썬은 자동으로 함수반환 값으로 None을 돌려준다. None은 NoneType이라는 타입에 속하며, 값이 없음을 표현하는 특별한 용도로 쓰인다(None이란 단어는 아무것도 없다는 뜻이다).

```
def say_name(kid): ---- 아이 이름을 문자열로 받음
    print(kid) ---- 명시적으로 아무것도 반환하지 않으면 파이썬이 None을 반환함
def show_kid(kid): ---- 아이 이름을 문자열로 받음
    return kid ---- 문자열 반환
say_name("몽룡") ---- 콘솔에 "몽룡"을 출력
show_kid("춘향") ---- 콘솔에 아무 것도 출력하지 않음
print(say_name("향단")) ---- 향단을 출력한 다음 None을 콘솔에 출력
print(show_kid("방자")) ---- 콘솔에 방자를 출력
```

코드 21-5
반환문이 있는 함수와 반환문이 없는 함수

21.2.2 return 문이 없는 함수

» 그림 21-2는 각 함수 호출이 반환하는 값이 어떤 대상을 대신하는지 보여준다.

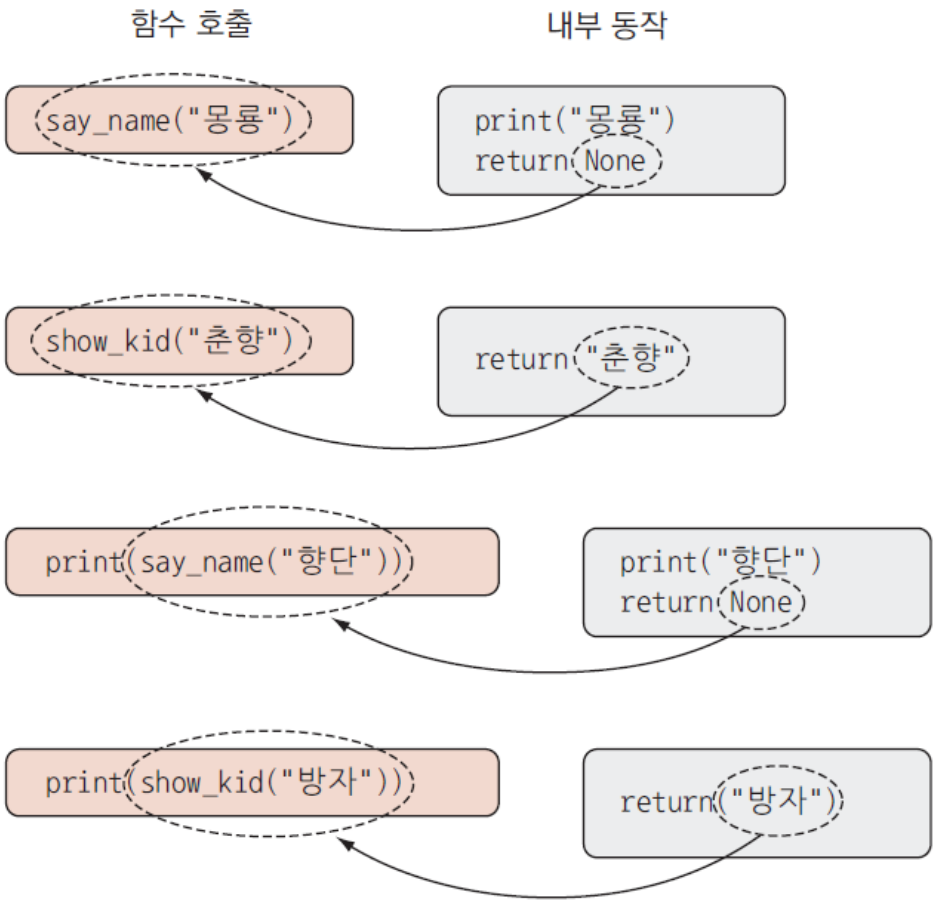

그림 21-2
네 가지 경우(값을 반환하지 않는 함수를 호출하는 경우, 값을 반환하는 함수를 호출하는 경우, 값을 반환하지 않는 함수를 호출한 결과를 출력하는 경우, 값을 반환하는 함수를 호출한 결과를 출력하는 경우)에 대한 그림.

왼쪽 상자는 함수 호출이며, 오른쪽 상자는 함수 호출 시 내부적으로 일어나는 동작이다.

함수에 return 문이 없으면 return None이 자동으로 맨 마지막에 추가된다. 검은 점선은 함수가 반환하는 결과에 의해 대치되는 값을 보여준다

21.2.2 return 문이 없는 함수

셀프 체크 21.7

```
def make_sentence(who, what):
    doing = who+" is "+what
    return doing

def show_story(person, action, number, thing):
    what = make_sentence(person, action)
    num_times = str(number) + " " + thing
    my_story = what + " " + num_times
    print(my_story)

who = "Hector"
what = "eating"
thing = "bananas"
number = 8
```

1. sentence = make_sentence(who, thing)
2. print(make_sentence(who, what))
3. your_story = show_story(who, what, number, thing)
4. my_story = show_story(sentence, what, number, thing)
5. print(your_story)

21.3 함수 문서화하기

21.3 함수 문서화 하기

함수는 코드를 모듈화하는 도구이기도 하지만, 코드 일부분을 덩어리째 추상화하는 방법이기도 하다.

- 인자를 받아서 처리하게 함으로써 함수를 여러 다른 상황에 더 일반적으로 사용할 수 있다.
- **함수 명세(specification)** 또는 **독스트링(docstring)**이라고 부르는 방식으로도 추상화를 달성할 수 있다.
- 독스트링을 읽으면 함수가 어떤 입력을 취하고, 어떤 역할을 수행하며, 어떤 값을 반환하는지 빠르게 알 수 있다.
- 독스트링을 읽는 것이 함수 구현 전체를 읽는 것보다 훨씬 빠르다.

```
def take_attendance(classroom_roaster, who_is_here):
    """
    classroom_roaster, tuple
    who_is_here, tuple

    classroom_roaster(출석부)에 있는 모든 사람이
    who_is_here(현재 교실에 있는 사람 목록)에 들어 있는지 검사한다
    학생이 who_is_here에 있는 경우에만 그 이름을 출력한다

    Returns "출석 부르기 끝"
    """
```

21.4 요약

21.4 요약

- 함수 정의는 그냥 함수를 정의할 뿐이다. 함수를 실행하려면 코드의 다른 부분에서 함수를 호출해야만 한다.
- 함수 호출을 그 함수가 반환한 값으로 대치할 수 있다.
- 함수는 객체를 하나만 반환한다. 하지만 튜플을 사용하면 둘 이상의 값을 한꺼번에 반환할 수 있다.
- 함수 독스트링은 함수 구현에 대한 자세한 사항을 추상화하고 문서화한다.

21.4 요약

» (Q21.1)

1. price라는 실수와 percent라는 정수를 매개변수로 받는 calculate_total이라는 함수를 작성하라. calculate_total은 percent로 지정한 비율만큼 팁을 추가해서 지불해야 할 전체 금액을계산해 돌려준다. 계산식은 total = price + percent * price이다.
2. 1에서 만든 함수를 호출하라. 인자로 price에는 20을 percent에는 15를 넘겨라.
3. 1에서 만든 함수를 사용하도록 다음 코드를 마무리하라.

```
my_price = 78.55
my_tip = 20
# 새로운 전체 금액을 계산하는 코드를 여기 넣는다
# 위에서 계산한 전체 금액을 출력하는 코드를 여기 넣는다
```