Function Design Recipe

(Revised)

1. **Header** Write the function header.

```
def is_even(num):
```

2. **Type Contract** Inside an indented triple-quoted string, write a type contract that identifies name and type of each parameter. Choose a meaningful name for each parameter. Also identify the return type of the function.

```
def is_even(num):
    """
    @type num: int
    @rtype: bool
    """
```

3. **Example** Write some examples of calls to your function[1] and the expected returned values. Include an example of a *standard* case (as opposed to a tricky or corner case.)

```
def is_even(num):
    """
    @type num: int
    @rtype: bool
    >>> is_even(2)
    True
    >>> is_even(17)
    False
    """
```

4. **Description** In the same line as the opening triple-quote mark, put a one-line summary of what the function does. If necessary, you can put an optional, longer description above the type contract. Mention each parameter by name.

```
def is_even(num):
    """Return whether <num> is evenly divisible by 2.
    @type num: int
    @rtype: bool
    >>> is_even(2)
    True
    >>> is_even(17)
    False
    """
```

[1] Do not include examples for functions that involve randomness or user input.

5. **Body** Write the body of the function by remembering to indent it to match the docstring. To help yourself write the body, review your example cases from step 1 and how you determined the return values. You may find it helpful to write a few more example calls in the docstring.

```
def is_even(num):
    """Return whether <num> is evenly divisible by 2.
    @type num: int
    @rtype: bool
    >>> is_even(2)
    True
    >>> is_even(17)
    False
    """
    return num % 2 == 0
```

6. **Test Your Function** Test your function on all your example cases including any additional cases you created in step 5. Additionally try it on extra *tricky* or *corner* cases.