CSC108 Recipe for Designing Functions

1. **Example** Write one or two 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.) Put the examples inside a triple-quoted string that you've indented since it will be the beginning of the docstring.

```
    """
    >>> is_even(2)
    True
    >>> is_even(17)
    False
    """
```

2. **Type Contract** Write a type contract that describes the types of the parameters and any return values.

```
    (str) -> int
    (str, bool) -> NoneType
    (list of int, tuple of (str,int)) -> list
```

   Put it on the same line as the opening triple-quote mark.

```
    """ (int) -> bool
    >>> is_even(2)
    True
    >>> is_even(17)
    False
    """
```

3. **Header** Write the function header above the docstring and outdent it. Choose a meaningful name for each parameter.

```
def is_even(value):
    """ (int) -> bool
    >>> is_even(2)
    True
    >>> is_even(17)
    False
    """
```

4. **Description** Before the examples, add a description of what the function does and mention each parameter by name.

```
def is_even(value):
    """ (int) -> bool
    Return True iff value is evenly divisible by 2.
    >>> is_even(2)
    True
    >>> is_even(17)
    False
    """
```

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.

```
def is_even(value):
    """ (int) -> bool
    Return True iff value is evenly divisible by 2.
    >>> is_even(2)
    True
    >>> is_even(17)
    False
    """
    return value % 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.

[1] Do not include examples for functions that involve randomness or I/O.

Another Example *Write a function that accepts the number of pizzas that you are ordering and the number of slices per pizza and returns the total number of slices in the order.*

1. **Examples**

```
"""
>>> total_slices(1, 8)
8
>>> total_slices(3, 12)
36
"""
```

2. **Type Contract**

```
""" (int, int) -> int
>>> total_slices(1, 8)
8
>>> total_slices(3, 12)
36
"""
```

3. **Header**

```
def total_slices(num_pizzas, slices_per_pizza):
    """ (int, int) -> int
    >>> total_slices(1, 8)
    8
    >>> total_slices(3, 12)
    36
    """
```

4. **Description**

```
def total_slices(num_pizzas, slices_per_pizza):
    """ (int, int) -> int
    Return the total number of slices in num_pizzas pizzas that each have
    slices_per_pizza slices.
    >>> total_slices(1, 8)
    8
    >>> total_slices(3, 12)
    36
    """
```

5. **Body**

```
def total_slices(num_pizzas, slices_per_pizza):
    """ (int, int) -> int
    Return the total number of slices in num_pizzas pizzas that each have
    slices_per_pizza slices.
    >>> total_slices(1, 8)
    8
    >>> total_slices(3, 12)
    36
    """
    return num_pizzas * slices_per_pizza
```

6. **Test** Call your function and compare the return values to what you are expecting.