CSC108H Lecture 23

Dan Zingaro

November 3, 2014

Testing with Doctest

To automatically test examples written in docstrings:

- `import` your module `m`
- `import doctest`
- `doctest.testmod(m)`
- Careful: trailing spaces in your example output can cause tests to fail
- Careful: put spaces between list elements: `[1, 2, 3]`

Two Functions

Let's discuss and write both of these functions.

```
def f1(lst):
  '''Return the first element from nonempty lst.
  '''

def f2(lst):
  '''Remove the first element from nonempty lst.
  '''
```

Example: Nested Lists

Add an example to the docstring. Write the function. Test the example automatically. Then give a full test suite.

```
def average_grade(grade_list):
  '''(list of list of [str, int]) -> float
  Return the average grade for all the students
  in grade_list. The inner lists of grade_list contain
  a student ID and a grade.
  '''
```

Example: Strings

Add an example to the docstring. Write the function. Test the example automatically. Then give a full test suite.

```
def choose_chars(s1, s2, mask):
  '''(str, str, str) -> str
  s1, s2, and mask are all of the same length.
  mask consists only of characters 0 and 1.
  Return a string where index i is s1[i] if
  mask[i] is 0 and s2[i] if mask[i] is 1.
  '''
```

Example: Dictionaries

Add an example to the docstring. Write the function. Test the example automatically. Then give a full test suite.

```
def to_dict(lst):
  '''(list of object) -> dict
  Return a dict whose key/value pairs are the pairs
  of elements in lst. lst has even length.
  '''
```

Returning and Modifying Dictionaries

- When a function returns a dict, do not hard-code the returned dict as the return value
  - Doctest compares strings, and we can't predict the string of the dict that will be returned
  - e.g. it could be `{1: 2, 3: 4}` or `{3: 4, 1: 2}`
- Instead, store the returned dict in a variable, and use == to compare that variable to the expected dict