CSC110 Lecture 4 Notes

Anatoly Zavyalov

September 21, 2020

1 The Function Design Recipe

Given a set of `int`s and an `int n`, return a mapping where each key is an `int` from the given set, and its corresponding value is a `bool` representing whether that `int` is divisible by `n`.

Here's how one approaches this problem:

1. **Write examples**: make sure you understand what the function is supposed to do.
2. **Write the function header**: formalize the function's name, parameters, and type contract.
3. **Write a description in English**: make sure you understand waht the function is supposed to do, and explain it precisely. Also, specify any additional requirements of the inputs
4. **Implement the function body**: write the code to make the function do what it's supposed to do
5. **Test the code!** Make sure you did the previous four steps right!

Here's what the function code might look right:

```python
def divisible_map(nums: set, n: int) -> dict:
    """Return a mapping from each given number to a boolean representing whether that number is divisible by n.

    Return a mapping where
    >>> result = divisible_map({1, 2, 20}, 2)
    >>> result == {1: False, 2: True, 20: True}
    True
    >>> result = divisible_map(set(), 9)
    >>> result == {} # Empty dictionary
    True
    """
    return {x: (x % n == 0) for x in nums}
```

2 Testing Your Functions

2.1 Doctest examples

```
"""
>>> result = divised_by({2, 3, 20}, 2)
>>> result == {2: True, 3: False, 20: True}
True
"""
```

Doctest examples are both **documentation** and **tests**.

2.2 Automatically running doctest examples

Include this code at the bottom of your Python file:

```
if __name__ == '__main__':
    import doctest # import the doctest library
    doctest.testmod() # run the tests
```

2.2.1 Warnings about doctest

1. If all tests **pass**, you won't see any output by default. However, you can call `doctest.testmod(verbose=True)` to get output on the doctests even if they pass.

2. Tests must be formatted correctly, identical to Python console.

3 Importing modules

3.1 Terminology

A file containing Python code (`.py`) is called a **module**.

A Python module can:

1. be run directly by the Python interpreter, or
2. contain code that is meant to be used by other Python modules (”library”)

3.2 Use code from another Python file

`import` statement:

```
import <module_name>
```

This statement:

1. Finds a Python module with the corresponding name.
2. Runs that file, storing all defined functions/data types in a new variable (the `<module_name>`).

3.3 Example

Importing the `math` module:

```
import math
```

This will allow the use of methods such as:

- `math.sqrt`
- `math.dist`
- `math.sin` (note that `math` trigonometric functions are in radians!)
- `math.asin`