COMPSCI 107
Computer Science Fundamentals

Lecture 08 – Documentation, debugging

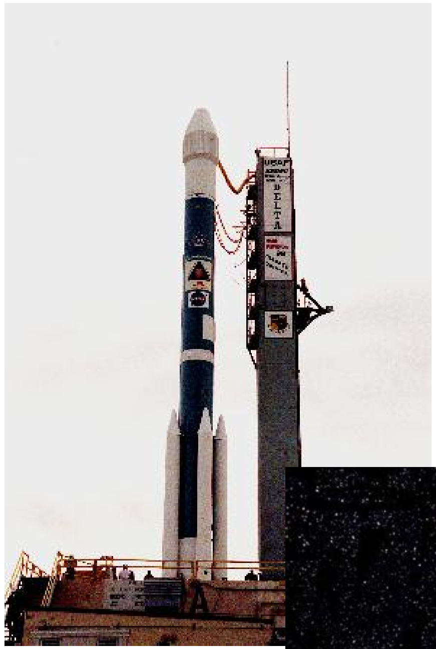
Documentation

- docstring
  - A special kind of string (text) used to provide documentation
  - Appears at the top of a module
  - Appears at the top of a function
  - Uses three double-quotes to surround the documentation
  - All modules , and all functions should include a docstring
  - Using the **help** function returns the docstring for that function

```
"""Converts a length in inches to a length in centimetres.

Author: Andrew Luxton-Reilly

"""
length_in_inches = 100

length_in_cm = length_in_inches * 2.54

print(length_in_cm)
```

Lost in Space (1999)

- The NASA subcontractor that built the Mars Climate Orbiter used English units instead of the intended metric system.
- This confusion led the Orbiter's thrusters to fire at the wrong time, causing it to crash on its arrival at Mars in 1999.
- This embarrassing error wasted $327 million, not to mention the year needed for the Orbiter to reach Mars.



Comments

- Comment
  - A programming comment is a note to other programmers
  - Anything between a # and the end of the line is ignored by the computer
  - Add comments sparingly to explain code that is difficult, or tell other programmers something they need to know about the code.

```
"""Converts a length in inches to a length in centimetres.

Author: Andrew Luxton-Reilly

"""
length_in_inches = 100    #Alter this value to convert a different length

length_in_cm = length_in_inches * 2.54

print(length_in_cm)
```

Exercises

- Improve the function that calculates the area of a circle by adding a docstring specifying the purpose of the code, the arguments and the return value.

Turning a Blind Eye (1991)

- During the 1991 Gulf War, the U.S. deployed its Patriot missile system to protect its troops, allies, and civilians from Iraqi SCUD missile attacks.
- A software rounding error in the system calculated time incorrectly, causing it to ignore some incoming missiles.
- A missile battery in Saudi Arabia failed to intercept an incoming SCUD that destroyed a U.S. Army barracks, killed 28 soldiers, and injured 100 others.

How do you know if your code works?

- Syntax errors
  - Easy to identify
  - Static analysis possible
  - The compiler tells you
- Runtime errors
  - Occur while the program is running
  - Provide feedback about when the program caused the error
  - Often harder to fix than syntax errors but easier than logic errors
- Logic errors
  - Difficult to identify
  - Program does exactly what you told it
  - Not always what you meant

Expensive Fireworks (1996)

- In 1996, code from the Ariane 4 rocket is reused in the Ariane 5, but the new rocket's faster engines trigger a bug in an arithmetic routine inside the flight computer.
- The error is in code to convert 64-bit floating-point numbers to a 16-bit signed integers. The faster engines cause the 64-bit numbers to be larger, triggering an overflow condition that crashes the flight computer.
- As a result, the rocket's primary processor overpowers the rocket's engines and causes the rocket to disintegrate only 40 seconds after launch.

Test cases

- Before you write the code, figure out what output you expect

- Example: Write a function that calculates the area of a triangle

$$area\ of\ a\ triangle = \frac{height}{2} \times base$$

Height	Base	Area
20	123	1230
2	2	2
0	10	0

Exercise

- Write a function that calculates the area of a triangle.
- How can you check if your code works correctly?

doctest

- doctest is a simple system for testing code
  - Not as sophisticated as unit testing
  - Tests are included in the docstring for a function
  - Any line that begins with the Python prompt >>> will be executed
  - The output from executing the code will be compared with the line following

```
>>> circle_area(1)
3.1415927
```

include this inside your docstring for the circle_area function

- To test the code, include the following statements in the module

```
import doctest
doctest.testmod()
```

Example

```
def triangle_area(base, height):
    """Returns the area of a triangle.

    Arguments:
        base -- a number (float) or (int) representing the length of the triangle base
        height -- a number (float) or (int) representing the length of the triangle height

    Returns:
        The area of the triangle (float)

    >>> triangle_area(10, 5)
    25.0
    >>> triangle_area(1, 1)
    0.5
    >>> triangle_area(2.5, 2)
    2.5
    """
    return base * height / 2
```

More doctest

- The output from the function is compared against the doctest output line by line until a blank line is encountered
  - SO if your output includes blank lines you need to use:
  - <BLANKLINE> in the doctest output

```
"""
>>> print_many_lines()
The first line
<BLANKLINE>
The last line
"""
```

Issues with doctests

- Testing with floating point values is tricky
  - Why? Use round(value, dp)
  - Round is tricky. Why?

- Testing with long output
  - Use a doctest directive (essentially option flags)

```
 >>> math.pi  # doctest: +ELLIPSIS
3.1415926...
```

- Misdirected confidence
  - Tests are only as good as you make them

Alternative use of doctest

- [illegible]

```
#This code can be in any module
#When you want to run the tests, simply load this module
import doctest
doctest.testfile('tests.txt')
```

```
This is the tests.txt file

It can contain any text, including working notes and other examples

Assume XXX is the name of a file (module), and get_seed_words is the name of the
function that you want to test

>>> from XXX import get_seed_words

>>> get_seed_words('This is a test.')
['This']

etc.
```

Choosing test cases

- Example: 'seed' words
  - Given a string of text, output a list containing the first word of each sentence.
  - A word is defined as a sequence of one or more characters separated by whitespace. A word may include punctuation.
  - A sentence consists of a sequence of words in which the last word terminates with a full stop, or the end of the string.

- In each of the following cases, state the number of words and sentences (spaces indicated with ^, tab with \T and newline with \n)?

```
abc              a.b.c...
abc.             a^b.^c.
^abc             a\Tb
abc.def          a.\nb.
abc.def          a.\nb.
abc^def          a.\T^\n^b^.
abc-def          ^
```

Exercise

- Define a set of test cases for a function that returns a list of seed words for a given string of text.

- def seed_words(text):

Program Development

- Start by thinking of the different kinds of input and the output
- Test Cases
- Work on the solution, keeping the test cases in mind
- Test your code after each development advance

Debugging

- Debugging and tracing code are closely linked skills

- To debug your code, you need to know:
  - what your code *should* produce
  - what your code *does* produce
  - why is there is a difference

Debugging

- Example:
  - seedBank += [wordBank[0]]
  - IndexError: list index out of range

```
def gsw(text):
    sentenceBank = []
    wordBank = []
    seedBank = []
    text.replace('!','.')
    text.replace('?','.')
    if (text[len(text)-1] == '.'):
        text = text[:-1]
    sentenceBank = text.split('.')
    for i in range(0,len(sentenceBank)):
        if (sentenceBank[i] != ''):
            wordBank = sentenceBank[i].split()
            seedBank += [wordBank[0]]
    return seedBank
```

Summary

- Comments are directed towards programmers and ignored by the computer
- Docstrings are used to convey the purpose of modules and functions
- Doctest is a simple system to automate the testing of code