



BIOE50010 – Programming 2

Computer Lab 9

Binghuan Li | Department of Chemical Engineering

[*binghuan.li19@imperial.ac.uk*](mailto:binghuan.li19@imperial.ac.uk)

Maria Portela | Department of Bioengineering

[*maria.s-marques-figueiredo-portela18@imperial.ac.uk*](mailto:maria.s-marques-figueiredo-portela18@imperial.ac.uk)

December 04, 2023

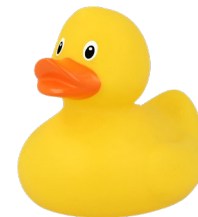
Coursework 2

- Will be released on *8th December (Friday)*, to be submitted 1 week after.
- Retrospectively, this coursework was designed in the *object-oriented programming* paradigm.
- But as you are aware, a solid mastery of the programming basics is *sine qua non* – re-visiting your labs and lectures over the entire term?
- Fear not – think of how far you have gone!
 - Very impressive (unexceptional) cohort performance in your coursework 1.

Questions should be logged on the

ed

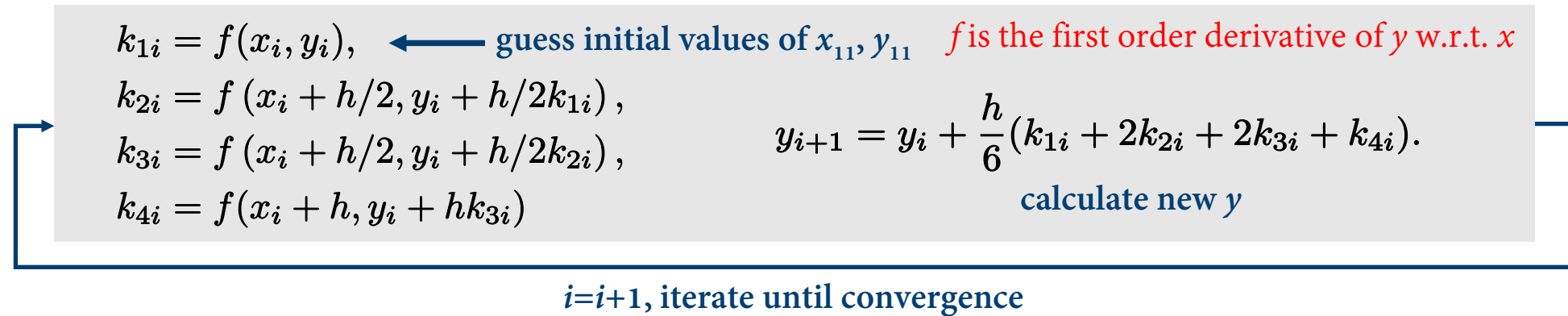
discussion



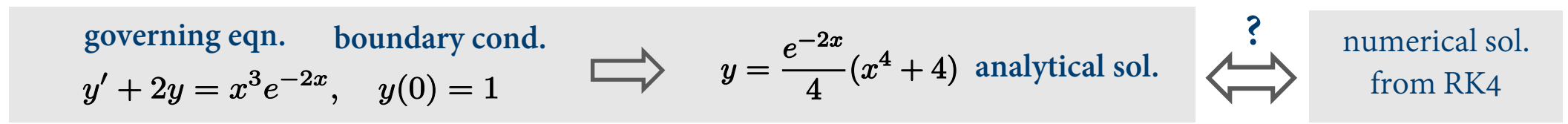
End of service time: 9 am on the submission day.

Why We Need Testing? One Example

- Suppose that you are implementing a *numerical* solver, using the 4th-order Runge-Kutta scheme, to solve a 2nd-order ordinary differential equation.



- However, you want to check whether your implementation is correct – hence, you come up with a test case, with the known *analytical* solution, to compare with the numerical solution. This is commonly referred to as the **sanity check**.



Unit Test (1/)

- Lucky us, Python provides a build-in package, **unittest**, for the testing purpose, *e.g.*, to check the numerical example.
- **unittest** requires that:
 - You put your tests into classes as methods
 - You use a series of special assertion methods in the **unittest.TestCase** class

Example from `test_point_pp.py`

```
import unittest
import point_pp as point

class TestPointPP(unittest.TestCase):

    def test_add(self):
        result = point.add([10, 2],[1, 7])
        self.assertEqual(result, [11, 9])
```

Access to the testing methods
defined in the **unittest** package

Method names should begin with
a keyword **test**

assertEqual method evaluates the
coherence between the input and output

Unit Test (2/)

- **unittest** methods at a glance:

unittest Method	Checks that...	unittest Method	Checks that...
<code>assertEqual(a,b)</code>	<code>a==b</code>	<code>assertIsNone(x)</code>	<code>x is None</code>
<code>assertNotEqual(a,b)</code>	<code>a != b</code>	<code>assertIsNotNone(x)</code>	<code>x is not None</code>
<code>assertTrue(x)</code>	<code>bool(x) is True</code>	<code>assertIn(a, b)</code>	<code>a in b</code>
<code>assertFalse(x)</code>	<code>bool(x) is False</code>	<code>assertNotIn(a, b)</code>	<code>a not in b</code>
<code>assertIs(a,b)</code>	<code>a is b</code>	<code>assertIsInstance(a, b)</code>	<code>isinstance(a, b)</code>
<code>assertIs(a,b)</code>	<code>a is b</code>	<code>assertNotIsInstance(a, b)</code>	<code>not isinstance(a, b)</code>
<code>assertIsNot(a, b)</code>	<code>a is not b</code>		

- also the `setUp` method and `tearDown` method allows you to config & post-processing your testing objects.

OOP resources

- W3 Schools: [Python Classes \(w3schools.com\)](https://www.w3schools.com/python/python_classes.asp)
- Tips: [8 Tips For Object-Oriented Programming in Python - GeeksforGeeks](https://www.geeksforgeeks.org/8-tips-for-object-oriented-programming-in-python/)
- [Review: Object-Oriented Design \(article\) | Khan Academy](https://www.khanacademy.com/topics/object-oriented-programming) (good for revising concepts – careful, examples are not Python!)
- [Object-Oriented Programming \(OOP\) in Python 3 – Real Python](https://realpython.com/object-oriented-programming/)

Efficiency and algorithms resources

- Concepts overview: [Introduction to Computation Complex Theory - GeeksforGeeks](#)
- Algorithm analysis (Computer Science): [Analysis of Algorithms | Big-O analysis - GeeksforGeeks](#)
- Algorithm analysis (Computer Science): [Asymptotic notation \(article\) | Algorithms | Khan Academy](#)