FIT1008 – Intro to Computer Science
Tutorial 5

Semester 1, 2018

Objectives of this tutorial

- To understand Big O notation.
- To be able to find the best and worst case time complexity for simple algorithms.
- To be able to determine whether a sorting method is stable.
- Learn how to use Exceptions.

Exercise 1

Consider the following algorithm:

```
1 def mystery(a_list):
2     n = len(a_list)
3     for i in range(n//2):
4       other = n - i - 1
5       a_list[i], a_list[other] = a_list[other], a_list[i]
```

- What does the algorithm do?
- What is the exact running time, as well as the Big O complexity?
- Is there a difference between best and worst case? Explain.

Exercise 2

Write a version of bubble sort that alternates left-to-right and right-to-left passes through the list. This algorithm is called *shaker sort*.

- What is best and worst case time complexity.
- Is this sorting method stable?

Exercise 3

What does the following snippet of code do? Discuss.

```
a = [0, 1]
try:
    b = a[2]
    print('that worked!')
except ValueError:
    print("no it didn't!")
```

Exercise 4

You can catch all exceptions by using a bare `except` statement, such as the one shown in the example below. However, this is in general acknowledged as a bad idea. Why do you think this is the case? And what would be the correct way to do it?

```
a = [0, 1]
try:
    b = a[2]
    c = int('foo')
    d = e
    f = 1/0
    print(1 + '1')
except:
    print("what happened!?")
```

Exercise 5

How can you set up a unit test to check that a specific Exception is being raised on a given input?