Week 8 Tutorial

COMP10001 – Foundations of Computing

Semester 2, 2025

Clement Chau

- Libraries
- Advanced Functions
  - Lambda, Map, Filter
- Types of errors
  - Syntax, Runtime, Logic Errors

Agenda

1. Week 8 Discussion – **Tutorial sheet** (~ 60 mins)
2. One-on-one Q&A (~ 50 mins)

8 (15/9)	Exception handling	Number systems, text encoding	Consolidation lecture, mid-semester test review	Week 8 tutorial sheet Week 8 tutorial solutions	• Ed worksheets 10 and 11 due (15/9 at 6 pm) • Project 1 due (19/9 at 6pm)

Ed worksheets 10, 11 due (15/Sep, Monday at 6 pm)
Project1 (15%) due (19/Sep, Friday at 6 pm)

Revision: Libraries

- A library is a **collection of code** designed to be reused in different programs
- Libraries tend to contain basic functionality for a particular specialized purpose
  - e.g. Maths, Generating Plots (Diagram) in Python
- Common ones we use in this subject:
  - **math** (for functions like **sqrt()**, **log10()**, **factorial()**, and constants like **pi**, **e**)
  - **collections** (for functions like **defaultdict()**)

sem1-2025 > week-8 > sqrt.py

```
1   print(sqrt(16)) # DOES NOT WORK
2
3   import math
4   print(math.sqrt(16)) # WORKS NOW
5
6   from math import sqrt
7   print(sqrt(16)) # WORKS NOW
8
9   from math import sqrt as square_root
10  print(square_root(16)) # WORKS NOW
11
```

Revision: defaultdict

sem1-2025 > week-8 > dict.py > ...

```
1  text = "Hello, World!"
2
3  freq = {}
4  for char in text:
5      if char in freq:
6          freq[char] += 1
7      else:
8          freq[char] = 1
9
```

Without defaultdict

sem1-2025 > week-8 > defaultdict.py > ...

```
1  from collections import defaultdict
2
3  text = "Hello, World!"
4
5  freq = defaultdict(int)
6  for char in text:
7      freq[char] += 1
8
```

With defaultdict

Revision: Lambda functions

sem1-2025 > week-8 > lambda.py > ...

```python
def last_char(s):
    return s[-1]

last_char = lambda s: s[-1]

```

Sort the list of animals according to their last character.

```python
animals = ['cat', 'dog', 'elephant', 'giraffe', 'hippo']
sorted_animals = sorted(animals, key=last_char)
sorted_animals = sorted(animals, key=lambda s: s[-1])
```

Revision: Map

sem1-2025 > week-8 > advanced-functions > map.py > ...

```python
nums = [1, 2, 3, 4, 5]
# Write a program to return squared, squared numbers of the list nums.
```

```python
# Without using map + lambda functions
squared = []
for num in nums:
    squared.append(num ** 2)
```

```python
# Using map + lambda functions
squared = list(map(lambda num: num ** 2, nums))
```

Revision: Filter

sem1-2025 > week-8 > advanced-functions > filter.py > ...

```python
nums = [1, 2, 3, 4, 5]
# Write a program to return a list of numbers greater than 2
```

```python
# Without using filter + lambda functions
greater_than_2 = []
for num in nums:
    if num > 2:
        greater_than_2.append(num)
```

```python
# Using filter + lambda functions
greater_than_2 = list(filter(lambda num: num > 2, nums))
```

Revision: Types of errors

- Syntax Errors
  - if statements without the colon (**:**) at the end.
  - Unmatched brackets (e.g. an opening bracket without a closing bracket)
- Runtime Errors
  - IndexError
  - TypeError
  - KeyError
  - ZeroDivisionError
  - NameError
  - AttributeError
- Logic Errors
  - Simply, when your program is **not getting the expected output**.
  - **Hardest to debug**, because the compiler shows no error feedback!

TuteSheet W8 – Exercises 1

1. Assign the value of the square root of 2 to `var` using three different methods, with each method using one of the following ways of import.

```
import math
from math import sqrt
from math import sqrt as square_root
```

```
import math
var = math.sqrt(2)
```

import the entire math library, with any methods or constants it contains

```
from math import sqrt
var = sqrt(2)
```

only import the `sqrt` method from the math library

```
from math import sqrt as square_root
var = square_root(2)
```

only import the `sqrt` method from the math library, and give it a "**nickname**" called `square_root`

TuteSheet W8 – Exercises 2

2. Rewrite the following with a default dictionary

```
my_dict = {}
for i in range(10):
    if i % 3 in my_dict:
        my_dict[i % 3].append(i)
    else:
        my_dict[i % 3] = [i]
```

```
{0: [0, 3, 6, 9],
 1: [1, 4, 7],
 2: [2, 5, 8]}
```

```
# cleaner, fewer lines, automatic handling
from collections import defaultdict

my_dict = defaultdict(list) #automatically creates a new empty list for any missing key
for i in range(10):
    my_dict[i % 3].append(i) # directly appends i to the list at that key

print(my_dict)
```

```
defaultdict(<class 'list'>, {0: [0, 3, 6, 9], 1: [1, 4, 7], 2: [2, 5, 8]})
```

TuteSheet W8 – Exercises 3

3. Given that `food_list = ["sushi", "pizza", "hot pot", "fish and chips", "burgers"]`, what are the outputs of the below snippets of code?

(a)
```
sorted(food_list, key=lambda x:x[-1])
```

['pizza', 'sushi', 'fish and chips', 'burgers', 'hot pot']

(b)
```
list(filter(lambda x: len(x.split()) == 1, food_list))
```

['sushi', 'pizza', 'burgers']

(c)
```
def price(food):
    return len(food) * 2
list(map(price, food_list))
```

[10, 10, 14, 28, 14]

TuteSheet W8 – Exercises 4

4. Find three out of four errors in the following programs. For each error, specify the line number, the error type (syntax/runtime/logic) and provide the corrected line of code.

(a)
```
2  def disemvowel(text):
3      """ Returns string `text` with all vowels removed """
4      vowels = ('a', 'e', 'i', 'o', 'u')
5      answer = text[0]
6      for char in text:
7          if char.lower() is not in vowels:
8              answer = char + answer
9      print(answer)
```

line 4; *logic/run-time* (if empty string); `answer = ''`

line 6; *syntax*;
`if char.lower() not in vowels:`

```
def disemvowel(text):
    """ Returns string `text` with all vowels removed """
    vowels = ('a', 'e', 'i', 'o', 'u')
    answer = ""
    for char in text:
        if char.lower() not in vowels:
            answer = answer + char
    return answer
```

TuteSheet W8 – Exercises 4

4. Find three out of four errors in the following programs. For each error, specify the line number, the error type (syntax/runtime/logic) and provide the corrected line of code.

```
1  def big-ratio(nums, n):
2      """ Calculates and returns the ratio of numbers
3      in non-empty list `nums` which are larger than `n` """
4      n = 0
5      greater_n = 0
6      for number in nums:
7          if number > n:
8              greater_n += 1
9          total += 1
10     return greater_n / total
11
12 nums = [4, 5, 6]
13     low = 4
14 print(f"{100*big_ratio(nums, low)}% of numbers are greater than {low}")
```

line 1; syntex; `big_ratio(nums,n):`

line 4; *logic/(run-time as well since it would cause error as total is undefinedl)*; `total = 0`

line 9; *logic; remove one level of indentation (outside if block)*

line 13; *syntax*; remove indentation

TuteSheet W8 – Exercises 4

4. Find three out of four errors in the following programs. For each error, specify the line number, the error type (syntax/runtime/logic) and provide the corrected line of code.

(b)

```
def big-ratio(nums, n):
    """ Calculates and returns the
    in non-empty list `nums` which
    n = 0
    greater_n = 0
    for number in nums:
        if number > n:
            greater_n += 1
            total += 1
    return greater_n / total

nums = [4, 5, 6]
    low = 4
print(f"{100*big_ratio(nums, low)}
```

```
def big_ratio(nums, n):
    """ Calculates and returns the ratio of numbers
    in non-empty list `nums` which are larger than `n` """
    total = 0
    greater_n = 0
    for number in nums:
        if number > n:
            greater_n += 1
        total += 1
    return greater_n / total

nums = [4, 5, 6]
low = 4
print(f"{100*big_ratio(nums, low)}% of numbers are greater t
```

TuteSheet W8 – Practice Programming

1. In this task we will play with numbers again. For example, the number 89 has two digits - 8 at the first position and 9 at the second position. If we raise each digit of 89 to the power of its position and take a sum: $8^1 + 9^2$, the answer is 89 itself! We call the number with this property a *cool number*. Another example of a cool number would be 598 because $5^1 + 9^2 + 8^3 = 598$.

   Write a function `get_cool_number(n)` that takes a positive integer n and returns the n^{th} cool number, starting from 1. That is, the first nine cool numbers are 1 to 9, then 10 is not cool, 11 is not cool ... we don't see another cool number (the 10^{th} one) until 89, then the 11^{th} cool number is 135, and so on. Therefore, your function `get_cool_number(10)` should return 89.

```
print(get_cool_number(1))  #1^1
print(get_cool_number(9))  #9^1
print(get_cool_number(10)) #8^1+9^2 = 8+81
print(get_cool_number(11)) #1^1+3^2+5^3 = 1+9+125
```

```
1
9
89
135
```

TuteSheet W8 – Practice Programming

```python
def num_is_cool(num):
    digit_sum = 0
    for i in range(len(str(num))):
        digit = int(str(num)[i])
        digit_sum += digit ** (i + 1)
    return digit_sum == num
```

```python
def num_is_cool(num):
    digit_sum = 0
    for index, digit in enumerate(str(num), start=1):
        digit_sum += int(digit) ** index
    return digit_sum == num
```

Raise digit to the power of (i + 1), and add the result
If the sum equals the original number → return True

```python
def get_cool_number(n):
    counter = 0
    current_number = 0
    while counter < n:
        current_number += 1
        if num_is_cool(current_number):
            counter += 1
    return current_number
```

If a number is cool, increase the count

When finding the n-th cool number → return it

Independent Work

- **Next due dates:**
  - Your **Project 1** is **due this Friday, September 19th, 6pm**.
    - For any questions, please go to the **First Year Centre 12pm-2pm every weekday** in Level 3, Melbourne Connect or ask in the **Ed Discussion** Forums!
    - You can also utilize the **PASS sessions** to get extra help.
    - We can only provide **very limited**, general guidance.
  - Ed Worksheets **12** and **13** is **due next Monday, September 22nd, 6pm**.

Scan here for annotated slides