COMP10001 Foundations of Computing
Semester 2, 2016

Tutorial Questions: Week 7

1. Identify 3 errors in the following code, determine for each whether it is a syntax, run-time or logic error, and provide a replacement line which corrects the error.

```
1  def disemvowel(string):
2      """Remove all vowels from 'string'"""
3      vowels = ('a','e','i','o','u')
4      answer = string[0]
5      for char in string:
6          if char is not in vowels:
7              answer = char + answer
8      print(answer)
```

A: *Any three out of:*

- *line 4; logic/run-time (if empty string);* `answer = ''`
- *line 6; syntax;* `if char not in vowels:`
- *line 7; logic;* `answer += char`
- *line 8; logic;* `return answer`

2. Identify 3 errors in the following code, determine for each whether it is a syntax, run-time or logic error, and provide a replacement line which corrects the error.

```
1  def big-ratio(numlist,n):
2      """Calculate the ratio of numbers in `numlist'
3      which are larger than 'n'"""
4      n = greater_n = 0
5      for number in numlist:
6          if number > n:
7              greater_n += 1
8              total += 1
9      return greater_n/total
10
11 nums = [4,5,6]
12     min = 4
13 print("{}% of numbers in {} are greater than {}"\
14       .format(100*big_ratio(nums,min),nums,min))
```

A: *Any three out of:*

- *line 1; syntax;* `def big_ratio(numlist,n):`
- *line 4; logic;* `total = greater_n = 0`
- *line 8; run-time (*`UnboundLocalError`*); fix through fix to line 4 as above*
- *line 8; logic; move outside the* `if` *statement (to the same indentation level as* `if`*)*

3. What does the following code do?

```
d = {}
for i in range(20):
    try:
        d[i%3].append(i)
    except KeyError:
        d[i%3] = [i]
print d
```

A: *The code generates a dictionary of lists of integers in the range 0 to 19 inclusive, with a list for all multiples of 3, one for all numbers with a remainder of 1 when divided by 3, and one for all all numbers with a remainder of 2 when divided by 3*

4. Rewrite the code from Question 2 using a `defaultdict`

```
from collections import defaultdict

d = defaultdict(list)
for i in range(20):
    d[i%3].append(i)
print d
```

5. What is wrong with the following code? Write test cases that will highlight the problems you identify, as well as two test cases that pass even though there is a problem.

```
  def contains(box,item):
      index = 1
      while index <= len(box):
          if box[index] == item:
              return True
          index += 1
      return False
```

A: *There are two problems: the code doesn't check the 0th element of* `box`*, and it also attempts to check the element with index* `len(box)`*, which will generate a run-time error*
Test case for first problem:

```
contains(['a','b','c'],'a')
```

(returns `False` *when it should return* `True`*)*
Test case for second problem:

```
contains(['a','b','c'],'d')
```

Test case which passes despite the problem:

```
contains(['a','b','c'],'b')
```

6. Write a code snippet to produce each of the following exceptions and associated error messages:

- `SyntaxError: invalid syntax`
  A: `1+`
- `ZeroDivisionError: integer division or modulo by zero`
  A: `1/0`
- `KeyError: 1`
  A: `d = {}; d[1] += 1`
- `IndexError: list index out of range`
  A: `lst = []; lst[1]`
- `TypeError: Can't convert 'int' object to str implicitly`
  A: `'a'+ 1`
- `ValueError: invalid literal for int() with base 10: 'a'`
  A: `int('a')`
- `TypeError: 'NoneType' object is not subscriptable`
  A: `[1,2,3].sort()[0]`
- `TypeError: object of type 'int' has no len()`
  A: `len(1)`

Optional Extension Questions for Self-study

1. The decimal number, $585 = 1001001001_2$ (binary), is palindromic (the same number read from the left or the right) as both a decimal (base 10) number and a binary (base 2) number.

   Find the sum of all numbers, less than one million, which are palindromic in both base 10 and base 2.

   Note that the palindromic number, in either base, may not include leading zeros.

   (adapted from Problem 36 of Project Euler)

   A:

```
def palnum():
    palsum = 0
    for i in range(1,10**6+1):
        deci = str(i)
        bini = bin(i)[2:]
        if deci == deci[::-1] and bini == bini[::-1]:
            palsum += i
    return palsum
```

2. The following iterative sequence is defined for the set of positive integers:

$$\begin{aligned} n &\to n/2 && \text{if } n \text{ is even} \\ n &\to 3n+1 && \text{if } n \text{ is odd} \end{aligned}$$

   The sequence terminates when it reaches 1.

   For example, starting from $n = 13$, we generate the following sequence:

$$13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$$

   which is of length 10 terms (including the starting value and the final value of 1).

   Which starting number(s), under one million, produce the longest chain? Optimise your solution to run to completion on IVLE, making use of redundancy in the calculation.

   (adapted from Problem 14 of Project Euler)

   A:

```
def collatz():
    max_len = 0
    max_vals = []
    cdict = {}
    for i in range(1,10**6):
        curr = i
        curr_len = 1
        while curr > 1:
            if not (curr % 2):
                curr /= 2
            else:
                curr = 3*curr + 1
            if curr in cdict:
                curr_len += cdict[curr]
                break
            curr_len += 1
        cdict[i] = curr_len
        if curr_len > max_len:
            max_len = curr_len
            max_vals = [i]
        elif curr_len == max_len:
            max_vals.append(i)
    return max_vals
```