CSC1010H Tutorial 5 – Mastering Computer Science, Testing, Debugging, and Python Functions

Introduction

Learning outcomes

Skills

- Be able to debug programs (i.e. methodically locate and repair errors) using tools such as Wing IDE and techniques such as executing tracing with print statements.
- Use Python functions and modules to apply a divide-and-conquer strategy to program development.
- Good studying and problem solving habits.

Knowledge

- Debugging: steps in the debugging process; Wing IDE debugging features (breakpoints, execution stepping, tracing); the use of print statements for the tracing of control and data flow; the commenting out of code to assist in bug location.
- Functions: concepts of function, parameter, and return value; form of function declaration; form of function calls; scope of variables; parameter passing (pass by value, pass by reference); modules; conditional program execution.
- Good studying and problem solving habits: features of a study plan (weekly, yearly).

Question 1 [25 marks]

Debugging is an important skill to help you understand your programs and fix them in a quick systematic way.

We have a program that needs debugging. (The original developers won the Lotto and have retired to Mauritius.) The intention is that this program serves as an automated technical support system. Users can enter technical queries and the system will suggest solutions.

Here's a sample of intended behaviour:

```
Welcome to the automated technical support system.
Please describe your problem:
My Computer CRASHED
Are the drivers up to date?
No
Curious, tell me more.
I don't have an internet connection
Contact Telkom.
I did
Curious, tell me more.
They told me to use Bluetooth
Have you tried mouthwash?
quit
```

CONTINUED

a) Using Wing IDE debugging facilities, find and fix the bugs in the program to ensure it works as illustrated above. *[15 marks]*
b) On the basis of this experience, write a short algorithm (a plain English report) for doing debugging using the Wing IDE.
Refer to actual features in Wing IDE, e.g. which buttons need to be clicked and which windows need to be viewed, etc. *[10 marks]*

The `technical_support.py` program can be found on the Vula assignment page.

Question 2 [25 marks]

Getting through university requires a great deal of hard work, planning and preparation. A year planner for each semester is useful to plan your time, primarily for anticipating and preparing for high pressure times.

Compile a year planner for this semester, including due dates for your assignments and test dates, for all your courses. Use appropriate colour coding for test and assignment dates.

Your submission can be in Microsoft Excel or pdf format.

Question 3 [25 marks]

On the Vula page for this assignment you will find a program called '`calendar.py`'. As the name suggests, it is a program for printing a calendar. Here's a sample of intended input and output:

```
Enter the year:
2015
                                2015

      January                February                March

Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su
          1  2  3  4                      1                      1
 5  6  7  8  9 10 11    2  3  4  5  6  7  8    2  3  4  5  6  7  8
12 13 14 15 16 17 18    9 10 11 12 13 14 15    9 10 11 12 13 14 15
19 20 21 22 23 24 25   16 17 18 19 20 21 22   16 17 18 19 20 21 22
26 27 28 29 30 31      23 24 25 26 27 28      23 24 25 26 27 28 29
                                              30 31

       April                  May                    June

Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su
       1  2  3  4  5                1  2  3    1  2  3  4  5  6  7
 6  7  8  9 10 11 12    4  5  6  7  8  9 10    8  9 10 11 12 13 14
13 14 15 16 17 18 19   11 12 13 14 15 16 17   15 16 17 18 19 20 21
20 21 22 23 24 25 26   18 19 20 21 22 23 24   22 23 24 25 26 27 28
27 28 29 30            25 26 27 28 29 30 31   29 30

        July                  August                September

Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su
       1  2  3  4  5                   1  2       1  2  3  4  5  6
 6  7  8  9 10 11 12    3  4  5  6  7  8  9    7  8  9 10 11 12 13
13 14 15 16 17 18 19   10 11 12 13 14 15 16   14 15 16 17 18 19 20
20 21 22 23 24 25 26   17 18 19 20 21 22 23   21 22 23 24 25 26 27
27 28 29 30 31         24 25 26 27 28 29 30   28 29 30
                       31

      October               November               December

Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su
          1  2  3  4                      1       1  2  3  4  5  6
 5  6  7  8  9 10 11    2  3  4  5  6  7  8    7  8  9 10 11 12 13
12 13 14 15 16 17 18    9 10 11 12 13 14 15   14 15 16 17 18 19 20
```

CONTINUED

```
19 20 21 22 23 24 25   16 17 18 19 20 21 22   21 22 23 24 25 26 27
26 27 28 29 30 31      23 24 25 26 27 28 29   28 29 30 31
                       30
```

The program is incomplete (those Lotto winning developers again). To work correctly, the program requires a Python module called '`calutils.py`', and we need you to write this.

The module should contain the following functions:

- `is_leap_year(year)`
  Given a year (a 4 digit number), returns true if it is a leap year, and false otherwise.
- `month_name(number)`
  Given the number of a month, returns the corresponding name. 1 is January, ..., 12 is December.
- `days_in_month(month_number, year)`
  Given a month (in the form of a number) and (4 digit) year, return the number of days in the month (accounting, in the case of February, for whether or not it is a leap year).
- `first_day_of_year(year)`
  Given a 4 digit year, return the number of the day on which January 1st falls. 0 is Sunday, …, 6 is Saturday.
- `first_day_of_month(month_number, year)`
  Given a month (in the form of a number) and (4 digit) year, return the number of the day on which the first of that month falls. 0 is Sunday, …, 6 is Saturday.

  In each case we've given the name of the required function and its parameters.

Useful tips

You may be surprised to find that you don't actually know how to determine if a year is a leap year. Did you know that 2000 was a leap year, but 1900 was not?

Given a 4 digit number representing a year, the day on which the 1st of January falls can be calculated using the following formula (Gauss's formula):

$$day = R(1 + 5R(Year - 1, 4) + 4R(Year - 1, 100) + 6R(Year - 1, 400), 7)$$

Where $R(n, m)$ is the remainder after dividing n by m e.g. $R(10, 3)$ is 1.

The formula produces a number in the range $0..6$, where Sunday is '=0', Monday is '1', and so on.

HINT: You might find it easier to accumulate the result by calculating the formula bit by bit. Let's say, for example, we wanted to calculate '$b = 2a^2 + 3a + 10$', we could write:

```
a=int(input('Enter a value for a: '))
b=2*a*a
b=b+3*a
b=b+10
print('The value of b is: ',b)
```

CONTINUED

Question 4 [25 marks]

This question concerns the construction of a Python module containing functions that may be used to make the computer output sounds representing Morse code transmissions. Since the automarker has no ears, and not everyone has headphones, the functions will simultaneously print a text representation of their behaviour.

Morse code

Morse code is a method of encoding text so that it can be communicated as a series of pulses of sound, light, or electricity. (e.g. http://en.wikipedia.org/wiki/Signal_lamp.)

There are two kinds of pulse: a dot, or a dash. A dash is three times the duration of a dot. Each letter of the alphabet is represented as a series of dots and dashes.

a	.−	g	−−.	m	−−	s	...	y	−.−−
b	−...	h	....	n	−.	t	−	z	−−..
c	−.−.	i	..	o	−−−	u	..−		
d	−..	j	.−−−	p	.−−.	v	...−		
e	.	k	−.−	q	−−.−	w	.−−		
f	..−.	l	.−..	r	.−.	x	−..−		

(We've used a period, '.', to represent a dot, and a hyphen, '-', to represent a dash.)

Pauses between pulses are an important:

1. The dots and dashes that form a letter code are separated by a period of silence equal the duration of a dot.

2. The letter codes forming a word are separated by a period of silence equal to the duration of 3 dots i.e. a dash.

3. Words are separated by a period of silence equal to the duration of 7 dots.

The task

Write a Python module called morse_code.py that contains the following functions:

1. `code_for(letter)`

   A function that returns the Morse code for a given letter ('a', ...,'z', 'A', ...,'Z'). For example, given the parameter value 'S', the function returns '...'.

2. `transmit_letter(letter)`

   A function that accepts a letter ('a', ...,'z', 'A', ...,'Z') as a parameter, determines the corresponding Morse code, and then makes the computer produce the required series of beeps and pauses.
   It also prints out text representing this behaviour:
   - when it makes a dot sound, it prints a '.',
   - when it makes a dash sound, it prints '-'.
   - when it pauses for a dot duration of time, it prints a '^'.

   For example, given a letter 'c', the code for which is '-.-.', the function will make the computer produce the sound sequence 'dash (*pause for duration of a dot*) dot (*pause for*

duration of a dot) dash (*pause for duration of a dot*) dot' and it will simultaneously print text '-^.^-^.'.

3. `transmit_word(word)`

   A function that accepts a word (composed of the letters 'a', ...,'z', 'A', ...,'Z'), determines the corresponding Morse code sequence, and then makes the computer produce the required series of beeps and pauses.
   It also prints out text representing this behaviour:
   - when it pauses for a dash duration of time between letters, it prints a '|'.

   For example, given 'tin', the code for which is '- .. -.', the function will make the computer produces the sound sequence 'dash (*pause for duration of a dash*) dot (*pause for duration of a dot*) dot (*pause for duration of a dash*) dash (*pause for duration of a dot*) dot' and it will simultaneously print the text '-|.^.|-^.'.

4. `word_pause()`

   A function that prints two vertical bars and causes the program to pause (be silent) for the duration of 7 dots.

NOTE: The 'transmit_word' function must use the 'transmit_letter' function, and the 'transmit_letter' function must use the 'code_for' function.

On the Vula assignments page you will find two Python files: '`morse_utils.py`' and '`morse_translator.py`'.

The `morse_utils` module contains a couple of functions that you should use for producing beeps and pauses:

1. `beep(frequency, duration)`
   Beep the PCs speaker at the given frequency (Hertz) and for the given duration (milliseconds).

2. `sleep(secs)`
   Suspend program execution for the given number of seconds.

Use a frequency of 1000 Hertz and a duration of 100 milliseconds for a 'dot' sound.

The `morse_translator` program uses the morse_code module that you will write. It asks the user to enter a sentence in English, translates it to Morse, and then outputs the appropriate sound sequence with accompanying textual representation.

Sample I/O:

```
Enter message:
Save our souls
.^.^.|.^-|.^.^.^-|.||-^-^-|.^.^-|.^-^.||.^.^.|-^-^-|.^.^-|.^-
^.^.|.^.^.
```

CONTINUED

Marking and Submission

Submit the (fixed) *techsupport.py*, debugging algorithm, year planner, *calutils.py*, and *morsecode.py* contained within a single .ZIP folder to the automatic marker. The zipped folder should have the following naming convention:

yourstudentnumber.zip

Marking Guide

Question 1 Tech Support + Debugging Report	25
Question 2 Semester planner	25
Question 3 Calendar module	25
Question 4 Morse code module	25
Total	**100**

Note that question 1 will partly be marked manually.

CONTINUED