COMP101: Introduction to Programming 2019-20

Assignment-05

Issue Date: **Wednesday 6th November 2019**

Submission Date: **Friday 15th November (noon)**

Summary:
Assignment-05 is worth 16% of the total marks for COMP101.
The assignment uses sequencing, selection, iteration, lists, function definitions and I/O control of strings and numbers.

You must submit an attempt at this assignment else a fail grade for the module will be awarded. There is an optional extended requirements you can attempt if you wish.

Submission details: Two files are required:
Submit one .py file AND one .pdf file

1) .py filename format:
familyName_givenName-CA05.py e.g. Smith_John-CA05.py

This file contains:
i) Your Python code edited in IDLE
The first 3 lines should be comment lines as follows:
#Your University id followed by Smith_John-CA05.py
#Month and Year of coding
#Brief description of the problem solved

AND

2) .pdf filename format:
familyName_givenName-CA05.pdf e.g. Smith_John-CA05.pdf

This file contains:
ii) A test table (use the test table template for evidence of testing)
iii) Pseudocode

Deadline Detail:

By the deadline indicated:

Your documents are to be submitted electronically via the department submission server at https://sam.csc.liv.ac.uk/COMP/Submissions.pl

Earlier submission is possible, but any submission after the deadline attracts the standard lateness penalties - see
http://www.csc.liv.ac.uk/department/regulations/practical.html

Plagiarism and collusion guidelines will apply throughout the assignment submission

COMP101 Assignment-05 2018-19

Assessment Information

Assignment Number	05 (of 07)
Weighting	16%
Assignment Circulated	As on front sheet
Deadline	As on front sheet
Submission Mode	e-submission
Learning outcome assessed	LO1: Identify principles and practice of using high level programming constructs to solve a problem LO3: Produce documentation in support of a programmed solution LO4: Use a suitable Integrated Development Environment to carry out implementation, interpretation/compilation, testing and execution LO6: Design and apply effective test cases
Purpose of assessment Marking criteria	Assessment of using sequence, selection, iteration constructs and defined functions to control I/O of strings and numbers in successful calculations for a given problem. Modularisation of code. Total marks over seven questions as a percentage
Submission necessary in order to satisfy module requirements?	Yes Assignments are not marked anonymously
Late Submission Penalty	Standard UoL Policy.

Requirements:
Design, implement and test a program using lists that manipulates strings and displays information based on the string manipulation.

No menu interface is required. Code using functions with argument/parameter passing, starting the code with a call to 'main()'

Problem Specification:
In film-making, directors, producers and casting staff have to think up character names for actors in the film. To help avoid this time-consuming problem, the problem domain lends itself to automation.

For the latest production of the next mega fantasy film 'Harry of the Rings', the casting producer has already chosen the character names for the leading actors.

But there are several support actors who need a name for their alien character. This can be automated by using the actors' real names and manipulating them to produce alien-sounding character names.

Taking each actor name in the form:

given_name family_name

to produce the following output after successful processing:
e.g.

```
Actor name:          john smith
Alien character:     smioj
```

Or

```
Actor name:          haoting wang
Alien character:     wanah
```

etc

1.Input the actor_name list as given – don't amend the names or the order they are in else you will alter the problem specification and won't be able to produce the expected output:
See the sample .py fragment later in this specification

2. Process each actor name for conversion to an alien name as shown in the examples: You need to store each alien name

3. When all alien names have been generated, look for are any alien names that do not have a vowel in them: Remove them from the list and store them separately

4. From the remaining alien names that have vowels in them extract the first three characters of each name

5. Concatenate these characters into a string and output the string produced followed by what the film is they present and banner list of the actor's names and a list of the alien names underneath this:
e.g.

<the string goes here> presents Harry of the Rings

Actor-list Alien –list

6. Take the alien names with no vowel in them, add the lowercase vowel 'a' to the end of each one:
Output it as a list of possible usable alien names under the above banner and under their own heading
e.g. alien name is smyakk, amend it to smykka and output the amended name

Output design and enhancement is at your discretion

NOTES:

1. Use the following data for the actor names: Don’t make any amendments to this source data and keep it in the order shown to avoid errors later on:

given_name	family_name
andrei	stephens
harry	venables
phillipa	blythe
yuan	spield
sadiq	elbahi
stephanie	wynne
zeng	ergan

2. FOR CONSIDERATION:
To keep us all together with the same data set (and to avoid deviations), you may wish to consider a call a function that will create the actor list.
It is easier if you let the program do this for you so you don’t have to input the list each time on execution.

Use the code fragment below in your code (or your own version)

```
def create_actor():

    actor_list = ["andrei stephens",

                  "harry venables",

                  "phillipa blythe",

                  "yuan spield",

                  "sadiq elbahi",

                  "stephanie wynne",

                  "zeng ergan"]


    return
```

A good scoring submission is one that addresses each step of the processing of the transition from actor name to alien-name as a set of discreet functions, passing parameters between them.

Mark scheme

Analysis and Design:	25%
Implementation:	50%
Quality of submission:	25%

Guidance:
Assessment is based on design, consideration of the problem domain, clarity, accuracy and appropriate use of code, testing and documentation.

The mark scheme looks for:
Appropriate constructs and data structures; modularisation with parameter handling, call to main()

Efficient use of variables to handle the input data and the use of theses variables to make the code clear and readable, thus aiding maintenance and debugging

Appropriate use of output techniques that are of benefit to the user

Use in-line comments sparingly but effectively

Appropriate testing to determine accuracy and/or problems (which may be documented in the test table comments column)

Note:
Because submission is handled electronically any file submitted past the deadline will be considered one day late. Late submissions are subject to the standard University Policy.