CE157 COURSEWORK AUTUMN 2023

Set by: Mike Sanderson

Deadline: 15 December 13:59:59

Introduction

You should refer to the Postgraduate Students' Handbook for details of the University policy regarding late submission and plagiarism; the work handed in must be entirely your own.

The assignment comprises six exercises. You should submit to FASER a single zip file containing (i) one .py file for each of the first 5 exercises, (ii) the output file produced by exercise 5 and (iii) a file in any standard text format for exercise 6.

Marking Criteria

Characteristics of an excellent project (70% or more):
• Excellent code documentation (i.e. comments using "#" in your Python scripts and documentation strings for all functions that specify previsely what the function does and returns and what the argjments are)
• Excellent use of Python's native methods, code standards, standard data structures and NumPy and/or Pandas dataframes where needed
• Excellent use of relevant data types
• Follows carefully the specification provided (where applicable)
• Excellent code optimisation in terms of result/outcome production and readability
• Generally, an excellent solution, carefully worked out, producing and demonstrating clearly and correctly results/outcomes
Characteristics of a good project (60%):
• Good code documentation
• Good use of Python's code standards, standard data structures and NumPy and/or Pandas dataframes where needed
• Good use of relevant data types
• Follows the specification provided (where applicable)
• Good code optimisation in terms of result/outcome production and readability
• Generally, a good solution which produces and demonstrates correct results/outcomes
Characteristics of a fair project (50% or less):
• No meaningful code documentation
• Code tends to be lengthier and more verbose than needed, or at times difficult to read
• No real thought on the relevance of data types
• Does not follow the specification provided (results/outcomes are produced without addressing the question).
• A solution that only seems to deliver part of the results without showing some logical code developments

If you submit solutions to only some of the exercises the mark awarded will be proportional to the percentage of the assignment that has been submitted

CE157 COURSEWORK AUTUMN 2023

Exercise 1 (8%)

Write a program that asks the user to input his/her date of birth in the format dd/mm/yyyy.

If the input is not in the correct format or the date is invalid the program should output an appropriate error message. If the input is a valid date the program should calculate and output the user's age. (You will need to check whether the user has had a birthday this year e.g. someone born on 01/11/2000 will be 23 on 15 December 2023, but someone born on 25/12/2000 will still be 22.) The program should additionally output a "Happy birthday" message if it is the user's birthday today.

To calculate the age you will need to obtain today's date; to do this you need to use the line

```
from datetime import date
```

and then use `date.today()` to obtain a `date` object containing today's date. You can access the day, month and year of a date `d` using `d.day`, `d.month` and `d.year` (these are all integer values).

Exercise 2 (12%)

Write a function that returns a list of the prime numbers between two positive integers supplied as arguments. Use this is in a program that asks the user to supply two positive integers, checks that the input is valid, then calls the function and outputs the numbers in the returned list, with 10 numbers per output line.

If the user enters negative numbers or supplies non-numeric input the program should output an appropriate error message; the two numbers should be accepted in either order. The range should be inclusive; if the user inputs 103 and 1009 (or 1009 and 103) these two numbers (which are both prime) should be included in the output.

CE157 COURSEWORK AUTUMN 2023

Exercise 3 (20%)

The three functions for this exercise should be written in a single .py file. You must not submit any code that calls the functions, although it is strongly recommended that you do produce such code in order to test your functions. The functions must not perform any input and output and should not raise any exceptions. The functions must have the names as specified so that I can test them using code that I will paste into your submissions.

a) Write a function called `fun1` that takes a string as an argument and returns `True` if and only if the string is a palindrome. The function should not be case sensitive, so "Dad" should be regarded as a palindrome. Spaces should be ignored so "red ER" should be regarded as a palindrome.

b) Write a function called `fun2` that takes a string as an argument, converts the string to lower-case and returns the third most frequent letter or digit. If there are equally frequent letters/digits you may return any one of the third most frequent; if there are fewer than 3 diferent letters or digits in the string `None` should be returned. Characters that are neither letters nor digits should be ignored.

c) Write a function called `fun3` that takes a string as an argument, counts the number of upper-case letters, lower-case letters, digits and punctuation symbols in the string and returns a tuple containing the four counts. (Any character that is not a letter, digit or space/tab/newline should be regarded as a punctuation symbol.)

CE157 COURSEWORK AUTUMN 2023

Exercise 4 (25%)

Write a function that takes 3 arguments, a list of tuples and two integers denoting the start and end of a salary range. Each tuple in the list will contain the name, job title and salary of an employee. The function should output the names and job titles of all employees in the list whose salary is greater than or equal to the first integer argument and also less than or equal to the second argument. The output should be displayed one employee per line, sorted by salary (largest first), in a neatly formatted table. If there are no matches an appropriate message should be output.

Write a second function that takes two arguments, a list of tuples of the same form as above and a string holding part of a job title. The function should output the names and job titles of all employees in the list whose job title contains the argument string.

Write a program that asks the user to supply a file name and attempts to open the file. Each line of the file should contain the name, job title and salary of a single employee (separated by commas). A typical line may be

```
Gareth Southgate,manager,2500000
```

The program should convert the contents of each line into a tuple with 3 elements and add the tuples to a list. (You may assume that the contents of the file are in the correct format so there is no need to perform validation). If the file cannot be opened the program should output an appropriate message and terminate.

After the input of the file contents has been completed the program should output the list of tuples (no formatting required) then enter a loop in which the user is asked which function he/she wishes to select. The user should then be asked to supply either start and end values for the salary range or part of a job title (depending on which function has been selected). The list of tuples and user input and should then be passed as arguments in a call to thechosen function described above. After returning from the function the user should be asked if he/she wishes to quit or to perform another search.

Hint: the elements of the tuple do not have to be stored in the order described above. A different order may make the sorting easier.

CE157 COURSEWORK AUTUMN 2023

Exercise 5 (25%)

The aim of this exercise is to process a file containing marks for a university module. The input file will have the format

```
5 3 30
1991111 65.5 45.5 47 29.5
2031234 75.5 68 49 51.5
2012345 33 50 48.5 50.5
2019734 55.5 51.5 72 61
2187345 39 47.5 51 38
```

The first line of the file contains the number of students, the number of pieces of coursework and the percentage coursework weighting, e.g. the 30 in this example denotes that the coursework comprises 30% of the overall module mark (with the exam making up the other 70%). It is assumed that each piece of coursework has the same weighting so that the overall coursework mark can be calculated as the average of the individual marks.

Each subsequent line contains a registration number, an exam mark and a set of coursework marks. (The number of coursework marks will be consistent with the number on the first line). The number of such lines will match the number of students.

Write a program that will open such a file, read the first line and create a 2-dimensional NumPy array where number of rows is the number of students and each row has 4 elements. (Initialise this array using a call to the `array` function with an argument `[[0,0.0,0.0,0.0]]*n`, where `n` is the number of students.)

The program should then read the remaining lines line-by-line and use the input values to store in a row of the array the registration number, the exam mark, the average coursework mark (as a real number – no rounding should be done at this stage) and the overall mark (calculated using the exam mark, the average coursework mark and the weighting, also as a real number). You may assume that all lines will contain the correct number of marks and the number of students will match the number specified on the first line.

Having generated the array, you should define a named data type (similar to `studType` on slide 46 of part 7 of the lectures slides) whose fields are 4 integers and one string. The program should then create a 1-dimensional array with `n` elements, using this named type as a `dtype` argument in the call to `np.array`. You could use a list containing multiple copies of a tuple with default values (e.g. 4 zeroes and an empty string) as the first argument.

From the data in each row of the first array the program should generate a structure containing the student's registration number, exam mark, coursework mark and total mark, all rounded to the nearest integer, and a grade string to be calculated using the rules below and store this in the corresponding element in the second array. (You can round a real number n to the nearest integer using `round(n)`.)

The program should then produce a version of the second array sorted by overall mark and output this array to a file (using a single call to the `print` function of the form `print(array2, file=f)`).

It should finally output to the screen/console the number of students who have distinction, merit and pass marks, the number of students who have failed and the registration numbers of the students who have failed.

You should submit to FASER the program file, and also the output file that your program produced from an input file that will be provided on Moodle by Tuesday of week 10.

Rules for calculating grades

Any student with a rounded mark of less than 35 for either the coursework average or the exam has failed, irrespective of the overall module mark. For all other students a rounded overall mark of 70 or more has a distinction grade, a rounded mark between 60 and 69 (inclusive) has a merit grade, a rounded mark between 45 and 59 has a pass grade, and a rounded mark below 45 fails.

Exercise 6 (10%)

The *selection sort* algorithm is described at https://www.javatpoint.com/selection-sort. The pseudocode in that link assumes that the `SMALLEST` function can set the value of an argument as a way to supply a result to the caller; in Python the only way to do this without writing a special class for the purpose is to return the result.

Produce a version of the pseudo code in which the algorithm is written as a function that takes a list/array as the only argument using the style and conventions presented in the lecture 8B slides (which can be found at the end of the lecture 8 section on Moodle); the `smallest` function should not take a `pos` argument and should return a result instead of setting `pos`.

Give an outline of what decisions need to be made when moving from the pseudocode to a Python program, stating with reasons what your choice would be.