CSSE1001 / 7030

Semester 1, 2017
Assignment 1
10 Marks

Due: Friday 31 March 2017 at 9:30pm

Introduction

Welcome to the dating game. With the popularity of *Married at First Sight*, we are going to build a matchmaking program. Some people worry that robots will replace workers, here is your chance to replace some pop-psychologists ☺.

The program will give the user a personality test and ask some questions about their physical characteristics and preferences. It will use these to find the best match in a database of potential partners.

You are provided with the code that reads the partner data from the database. You need to implement the functionality to perform the personality test, question the user and perform the matching. The database is in the form of a text file. You can edit the text file to add more partners. You must follow the text file format exactly as specified.

Example Interaction

The following is an example of one user using the system and getting their prospective match from the database. Your program needs to duplicate the demonstrated functionality and interaction. Your program may vary how it presents its output, but a tutor should be able to execute your program **without** looking at the screen and just typing in the required inputs.

```
Welcome to PyMatch

Please enter your name: Ian Foo

Hi Ian Foo.
What is your gender?
  1) male
  2) female
  3) other
Please enter your answer: 1

What is your sexual preference?
  1) male
  2) female
  3) other
Please enter your answer: 2

What is your height?
  1) tall
  2) medium
  3) short
Please enter your answer: 1
```

```
What height do you prefer your partner to be?
  1) tall
  2) medium
  3) short
Please enter your answer: 2

We will now ask you some questions to try to determine your
personality type.

Do you find it easy to introduce yourself to other people?
  1) Yes
  2) Most of the time
  3) Neutral
  4) Some times
  5) No
Please enter your answer: 4

Do you usually initiate conversations?
  1) Yes
  2) Most of the time
  3) Neutral
  4) Some times
  5) No
Please enter your answer: 4

Do you often do something out of sheer curiosity?
  1) Yes
  2) Most of the time
  3) Neutral
  4) Some times
  5) No
Please enter your answer: 2

Do you prefer being out with a large group of friends rather than
spending time on your own?
  1) Yes
  2) Most of the time
  3) Neutral
  4) Some times
  5) No
Please enter your answer: 2

Thank you for answering all the questions. We have found your best
match from our database and hope that you enjoy getting to know
each other. Your best match is:
Mary Smith
```

The final output of your program **must** be only the full name of the match on a line by itself. **Do not** provide any embellishments in the output as this will cause automatic program checking to **fail**.

Design

The focus of this assignment is on implementing and testing the required functionality. A simple design is provided below.

You should have one or two functions that prompt the user to answer the physical characteristics questions and the personality test questions. Some people will find it easier to implement this functionality with two functions.

```
physical_characteristics_question(question, answer1, answer2, answer3)
```

This function would take the question and three answer options as parameters. After displaying the question and answer options it will prompt the user for an answer choice. It will return the number corresponding to the answer that the user selected.

```
personality_question(question)
```

This function would take the question as a parameter. After displaying the question and the five standard answer options it will prompt the user for an answer choice. It will return the number corresponding to the answer that the user selected.

These two functions could be implemented as a single function, which is a better design but requires some knowledge of lists in Python. Implementing this function, rather than the two above, is optional. You can achieve full marks for the assignment with two functions like the ones described above. Implementing this single function would contribute towards **part** of the bonus mark.

The personality test scoring is a simple summation of two times the number corresponding to the user's answer to each question. For example, if a user answered the four personality test questions above as: 2, 2, 3 and 4; their personality test score would be (2 + 2 + 3 + 4) × 2, which equals 22. A personality match is determined by simply finding the potential partners with a test score as close as possible to the user's score. For example, if there were three potential partners with scores of 10, 12 and 40, then the person with the score of 12 would be the closest match for the user with a score of 22.

You should have at least one function that performs the matching. The matching algorithm guarantees that the genders and gender preferences of the user and suggested partner match. It then finds the potential partners in the database that have personality test scores that are closest to the user's score. From these potential partners, it then finds the best match based on the user's, and secondarily the potential partners', height preferences. If there is more than one best match it will just return the name of the first one found. If there is no match it returns `None`.

```
match(gender, sexual_pref, height, height_pref, personality_score)
```

The five parameters are the user's details. It uses the utility functions in the **`partners.py`** file to iterate through the database of potential partners and perform the matching logic. (An example of iterating through all of the potential partners in the database and accessing each person's details is provided in the file: **`partners_example.py`**.) You are not required to use the utility functions in the **`partners.py`** file but are encouraged to do so to reduce the amount of work you need to do to complete this assignment. (**Note:** The two classes in the **`partners.py`** file could be designed better. They have been simplified for use in this assignment, as you are not required to understand object-oriented concepts. Redesigning these classes will **not** contribute towards the bonus mark.)

The suggested functions above are the minimum requirements for implementing this program. You may implement more functions to provide better structure for your program's design. Consequently it is **not** required that your program have functions with the exact names and parameters described above.

Ideally, your program should not have any global variables and should have a main function that drives your program. Failing to achieve this will result in losing 1 or 2 marks for the programming constructs criteria.

When reading input from a user you should perform some simple checks to make sure the input is valid. Tutors will not perform esoteric input testing but if your program does not correctly catch invalid input such as '0' or 'a' you will lose some marks.

It is good design practice, and a good assessment strategy, to implement a program in stages. Doing this means that you are much more likely to have a working program to submit even if you cannot implement all of the requirements. As a suggestion, you should first implement the **`personality_question`** function and the **`match`** function that only uses the personality score to match a user to a partner. Next try implementing the **`physical_characteristics_question`** function and extend the **`match`** function to add in gender and gender preferences to the matching algorithm. Then extend the **`match`** function to add in height and height preferences to the matching algorithm. Finally add input testing to your program.

Database

The database is a text file containing the names and results of the PyMatch questions. Each line of the text file corresponds to a single person. Each data item on the line is separated by a single space. The first data item is the person's first name. The second item is their last name. The third item is their gender. The fourth item is the gender of the partner they are seeking. The fifth item is the person's height. The sixth item is their preference for the height of their partner. The seventh item is their personality test score. The structure of the text file is as follows:

```
first_name last_name gender gender_preference height heigh_preference personality_test_score
```

You should add new entries to the text file to test your program thoroughly.

Submission

You must submit your completed assignment electronically through Blackboard.
For information on submitting through Blackboard, please read:

https://web.library.uq.edu.au/library-services/it/learnuq-blackboard-help/learnuq-assessment/blackboard-assignments

You should submit your assignment as a single Python file called **`pymatch.py`** (use this name – all lower case). You do **not** need to submit the **`partners.py`** utility file or the database text file. You may submit your assignment multiple times before the deadline – only the last submission will be marked.

Late submission of the assignment will **not** be accepted. In the event of exceptional personal or medical circumstances that prevent you from handing in the assignment on time, you may submit a request for an extension.

See the course profile for details of how to apply for an extension:

http://www.courses.uq.edu.au/student_section_loader.php?section=5&profileId=85405

Requests for extensions **must** be made no later than 48 hours prior to the submission deadline. The application and supporting documentation (e.g. medical certificate) **must** be submitted to the ITEE Coursework Studies office (78-425) or by email to enquiries@itee.uq.edu.au. If submitted electronically, you **must** retain the original documentation for a minimum period of six months to provide as verification should you be requested to do so.

Assessment and Marking Criteria

This assignment assesses course learning objectives:

1. apply program constructs such as variables, selection, iteration and sub-routines,
3. read and analyse code written by others,
5. read and analyse a design and be able to translate the design into a working program,
6. apply techniques for testing and debugging

Criteria	Mark
Programming Constructs • Program is well structured and readable • Variable and function names are meaningful • Appropriate constructs are employed • Functions correspond to design expectations • Algorithmic logic is appropriate	 1 1 1 1 2
Sub-Total	6
Functionality • Implements correct functionality with no serious errors • Implements correct functionality with some errors • Implements part of the functionality	 2 1.5 0.5
Sub-Total	2
Documentation • Entire program is documented clearly and concisely, without excessive or extraneous comments • Program is documented clearly, with all functions having meaningful docstring comments • Some parts of the program have adequate comments	 2 1.5 0.5
Sub-Total	2

Bonus
• Improved on the provided design. This bonus will only be awarded if the total mark for the previous sections is at least 7.5. Changing the design, without improving it, will result in a low mark for the Programming Constructs section.
• Criteria that will be used for assessing if the design has been improved are:
  ○ simplification of logic (e.g. combining the two question functions into a single function); and
  ○ improving the cohesion of functions (cohesion means that all parts of the function relate to a **single** purpose).
• The **maximum** possible mark for this assignment is 10, even if the total for the assignment and the bonus is greater than 10.
	1

Total	**/ 10**

In addition to providing a working solution to the assignment problem, the assessment will involve discussing your code submission with a tutor. This discussion will take place in week 6, in the practical session you have signed up to. You **must** attend that session in order to obtain marks for the assignment.

In preparation for your discussion with a tutor you may wish to consider:

- any parts of the assignment that you found particularly difficult, and how you overcame them to arrive at a solution;
- whether you considered any alternative ways of implementing a given function;
- where you have known errors in your code, their cause and possible solutions (if known).

It is also important that you can explain to the tutor how each of the functions that you have written operates (for example, if you have used a for loop or a while loop in a function, why this was the right choice).

Marks will be awarded based on a combination of the correctness of your code and on your understanding of the code that you have written. **A technically correct solution will not achieve a pass mark unless you can demonstrate that you understand its operation.**

A partial solution will be marked. If your partial solution causes problems in the Python interpreter please comment out the code causing the issue and we will mark that. Python 3.6 will be used to test your program. If your program works correctly with an earlier version of Python but does not work correctly with Python 3.6, you will lose **at least** all of the marks for the functionality criteria.

Please read the section in the course profile about plagiarism. Submitted assignments will be electronically checked for potential plagiarism.