Foundation

Advanced Python

Project paper 1

Deadline: 1pm on Friday of the 2nd week

Version	Date	Author	Comments
1.0	28 / 01 / 22	Nikola Ignjatovic	First draft

About the project

1.1 Structure

The project contains 15 questions.
The deadline for submission is 1pm on Friday of the 2nd training week.

1.2 Marking criteria

Each question will be marked according to the following criteria:

- Does it work?
  Where possible, this will be checked by running a series of test cases. All tests must pass to receive the max available percentage. You will be given a full set of test cases so that you can check your code before submitting it.
  Tests passed by use of hard coding will be deemed to have failed.
  Where unit testing is not applicable (i.e. plots), you will be expected to produce the carbon copy of the plot as in the provided image within the question and using the module requested in the question (Matplotlib or Seaborn).
  Although unit tests will check the correctness of every class functionality, you may find it easier to write some client code to check that everything works correctly; just ensure all client code is commented out before running the unit tests and is removed before submission
- Is it written using clean code?
  This will be checked against the clean code rules stated below

1.3 Clean code

Clean code is part of "good programming practice"; please follow all clean code rules outlined below to avoid mark reductions

- General rules:

- All your code must be written only in provided designated files named q1.py, q2.py, ..., q15.py (qx.py, x = 1 to 15); only code in these files will be looked at when assessing your work
- Do not change any existing text within the scripts (qx.py, x = 1 to 15); where requested to write a function, the function name specified in the question will be identical to the function name in the script corresponding to that question; if changed it will fail the unit tests
- Files named qx-test.py (x = 1 to 15) are unit test files; they will help test your function against multiple test cases and must not be changed
- Do not change the name of the script files qx.py (x = 1 to 15), as it will prevent using the corresponding file qx-test.py (x = 1 to 15) to test your script

- Naming standards:
    - Variable/constant names should be clear and descriptive with no ambiguity; do not use one-letter variable/constant names
    - class names should follow UpperCamelCase convention
    - constant names should follow UPPER_WITH_UNDER convention
    - all other variable names should follow lower_with_under convention
- Use correct spacing and indentation
- Comments
    - Use of comments to explain your code is mandatory for every task
    - Use of comments should not be excessive
    - No commented-out and/or obsolete lines of code
- No print statements in any of the scripts unless requested by the task
- No input statements in any of the scripts
- No code which has no impact on the return value
- No excessively complex code; think about breaking code down into smaller, simpler functions if it gets too complex

1.4 Mark scheme

- Question 1 (15%)
    - Passes all tests (13%)
    - Clean code (2%)

- Question 2 (15%)
    - Passes all tests (13%)
    - Clean code (2%)
- Question 3 (15%)
    - Passes all tests (13%)
    - Clean code (2%)
- Question 4 (15%)
    - Passes all tests (13%)
    - Clean code (2%)
- Question 5 (15%)
    - Passes all tests (13%)
    - Clean code (2%)
- Question 6 (2%)
    - Passes all tests (1%)
    - Clean code (1%)
- Question 7 (3%)
    - Passes all tests (2%)
    - Clean code (1%)
- Question 8 (3%)
    - Passes all tests (2%)
    - Clean code (1%)
- Question 9 (3%)
    - Passes all tests (2%)
    - Clean code (1%)
- Question 10 (3%)
    - Passes all tests (2%)
    - Clean code (1%)
- Question 11 (3%)
    - Passes all tests (2%)
    - Clean code (1%)
- Question 12 (2%)
    - Passes all tests (1%)
    - Clean code (1%)
- Question 13 (2%)

    - Passes all tests (1%)
    - Clean code (1%)
- Question 14 (2%)
    - Passes all tests (1%)
    - Clean code (1%)
- Question 15 (2%)
    - Passes all tests (1%)
    - Clean code (1%)

- Pass (75-79%)
- Merit (80-89%)
- Distinction (90% +)

Note: Questions 1 – 5 are for pass. The first five with any combination of the remaining questions that adds up to at least 80% achieves merit; the first five with any combination of the remaining questions that adds up to at least 90% achieves distinction.

The questions

Question 1 – Lambda Function (pass)

In the file `q1.py` write a lambda function called `replace_ch_in_pos` which returns a string obtained from an original string by replacing a character in the specified position with the specified character.

Arguments:

- A non-empty string consisting of any number of any characters
- A string containing one character (the new character to replace an existing character in the non-empty string)
- A non-negative integer in the range between 0 and length of the string - 1 (containing the position at which the existing character needs to be replaced with the new character).

Return value:

- A string with the character at the specified position replaced by the given character

For example:

- 'ABCDE', '#', 3 should return 'ABC#E'
- 'Trainer', 'e', 6 should return 'Trainee'
- 'python', 'P', 0 should return 'Python'
- '12-11-2003-17:34:54', '-', 10 should return '12-11-2003 17:34:54'
- '12345', '0', 4 should return '12340'
- 'stutter', 'c', 1 should return 'scutter'
- 'P', '£', 0 should return '£'

Note: do not perform validation on any of the parameters. This is the responsibility of the function caller outside the lambda function.

Question 2 – Object-oriented (pass)

In the file `q2.py` create the class `Trainer` that includes the following:

- the class attribute count_trainers (int, set to 0), to store the number of trainers
- the constructor, accepting the following instance parameters:
  - trainer_id (int)
  - first_name (str)
  - last_name (str)
  - date_joined (str)

  to initialize the above 4 and the below 4 additional instance attributes:
  - email (str)
  - date_left (str) – set to None
  - courses (list, set to empty list), to store all courses taught by the trainer
  - trainings (dictionary, set to empty dictionary), to store all training deliveries by the trainer in the format date: (course, group)

  and to increment the class attribute count_trainers.
  The email address should be constructed from the lowercased first_name joined with the lowercased last_name by the dot (.) character in between, followed by @fdmgroup.com.
- the class method print_count() to print out the number of trainers, i.e. "The number of trainers is 1"
- the instance methods:
  - assign_course() to assign a course to the list of courses taught by the trainer
  - assign_training(), to assign a training to the trainer. It accepts three parameters: date(str), course (str) and group (str), checks whether the course exist in trainer's courses list and if so adds the training to the dictionary trainings in the form of date: (course, group) if the date does not already exist as a key. If it does, it changes the existing tuple (course, group) for the given date to the new (course, group) tuple. If the course does not exist in the trainer's courses list the following message is displayed:
    "The course <course> does not exist in the list of courses for the trainer <trainer_id> <first_name> <last_name>"

- terminate_employment() to be called for every trainer that leaves FDM. It assigns the date passed to it as argument to the instance attribute date_left and decrements the class attribute count_trainers.

Note: unit tests cannot test the correctness of methods that print out text; upon running the script q2-test, such text should be displayed in blue for you to check.

Question 3 – NumPy (pass)

In the file `q3.py` write a function called `pad_arr_int_to_str` to convert all elements of an original NumPy array of non-negative integers into a NumPy array of strings where each string is of equal length – the number of digits in the largest integer in the original array, and is composed of the integer value padded on the left with zeros.

Arguments:

- An array consisting of non-negative integers

Return value:

- An array of strings where each string is a zero-representation of the integer, padded to left with zeroes to make the length of each string the same as the number of digits of the largest integer.

For example:

- The array [45, 4832, 123, 987655] should return the array [‘000045’, ‘004832’, ‘000123’, ‘987655’]
- The array [529, 0] should return the array [‘529, ‘000’]
- The array [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] should return ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
- The array [0] should return the array [‘0’]
- An empty array should return an empty array

Question 4 – Pandas (pass)

The provided data set “fdm_training_data_2021.csv” contains details about all exams taken in every FDM academy worldwide in 2021.
Using the data from the provided input file “fdm_training_data_2021.csv”, in the file `q4.py` write the function called `avg_percentage_per_activity` to show the average percentage for each activity (listed in the 'Activity Name' column) for a specified course code after the specified exam attempt.
Important: Use the `'try'` statement to ensure safe reading from file, so that the script does not end up crashing if the file does not exist in the current working directory or if the file exists but includes some unreadable characters. Use the provided file “existing_unreadable_file” included to the exam files to test your code. In case the input file is not found, the following message should be returned (not displayed): The file “inexistent_file” does not exist. In case the input file is found, but is not readable, the following message should be returned (not displayed): Error reading file "existing_unreadable_file".

Arguments:

- A string storing the file name containing the data set. The file is expected to be found in the same folder where the script q4.py, containing the function definition is stored
- A string storing the course code for which to calculate averages
- A positive integer storing the attempt number

Return value:

- The DataFrame consisting of one row for each Activity Name within the specified course code (the number of rows coincides with the number of activities within the specified course code) and 4 columns (Course Code, Activity Name, Attempt, Average).
- The string containing the message: The file “filename” does not exist, in case the input file “filename” is not found in the current directory
- The string containing the message: Error reading file "filename", in case the input file “filename” contains unreadable character(s)

For example:

- For the file 'fdm_training_data_2021.csv', course code 'L-21-FOU-02' and attempt 1, the function should return the following DataFrame:

```
  Course Code                                Activity Name  Attempt   Average
0  L-21-FOU-02           Core - Business Fundamentals Exam        1  0.884615
1  L-21-FOU-02   Core - Business Fundamentals Presentation        1  0.839231
2  L-21-FOU-02         EMEA Pro Skills I/V Assessment 2021        1  0.901793
3  L-21-FOU-02  EMEA Pro Skills Presentation Assessment 2021     1  0.929757
4  L-21-FOU-02       EMEA Pro Skills Written Assessment 2021      1  0.926186
5  L-21-FOU-02                     EXCEL Exam - March 2017        1  0.822143
6  L-21-FOU-02                  EXCEL Project - March 2017        1  0.787857
```

- For the same file and course code arguments as above and attempt 2, the function should return the following DataFrame:

```
  Course Code                      Activity Name  Attempt  Average
0  L-21-FOU-02  Core - Business Fundamentals Exam        2     0.95
1  L-21-FOU-02            EXCEL Exam - March 2017        2     0.88
2  L-21-FOU-02         EXCEL Project - March 2017        2     0.78
```

- For the same file and course code arguments as above and attempt 3, the function should return an empty DataFrame, as there was no 3rd attempt:

```
Empty DataFrame
Columns: [Course Code, Activity Name, Attempt, Average]
Index: []
```

Note: To pass unit testing:

1. the order and spelling of the column names in the returned DataFrame must match the order and spelling of the column names in the above three examples: Course Code, Activity Name, Attempt, Average.
2. the two messages to be displayed in case the input file is not found in the current directory, and in case the input file contains unreadable character(s), must be identical to the ones shown above in red (with “filename” replaced with the file name being tested and stated within double quotes).

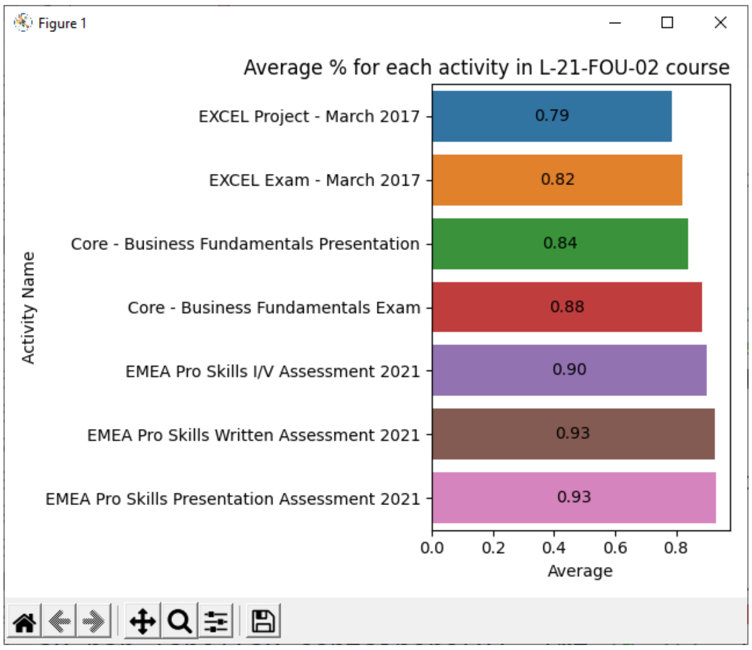
Question 5 – Seaborn (pass)

Using the data from the provided input file “fdm_training_data_2021.csv”, in the file `q5.py` create a **Seaborn bar plot** showing the average percentage for the 1st attempt on each activity in L-21-FOU-02 course. Sort the average percentage values in ascending order and include values for the bars rounded to 2 decimal places in the centre of the bars.
Draw the plot in **one** of the following two ways:

1. If you have created the function `avg_percentage_per_activity()` for Question 4, import it and use it to generate the data for the plot
2. If you have not created the function `avg_percentage_per_activity()`, create the DataFrame using the data from the first example given for Question 4:

```
   Course Code                                  Activity Name  Attempt   Average
0  L-21-FOU-02              Core - Business Fundamentals Exam        1  0.884615
1  L-21-FOU-02      Core - Business Fundamentals Presentation        1  0.839231
2  L-21-FOU-02             EMEA Pro Skills I/V Assessment 2021        1  0.901793
3  L-21-FOU-02   EMEA Pro Skills Presentation Assessment 2021        1  0.929757
4  L-21-FOU-02        EMEA Pro Skills Written Assessment 2021        1  0.926186
5  L-21-FOU-02                        EXCEL Exam - March 2017        1  0.822143
6  L-21-FOU-02                     EXCEL Project - March 2017        1  0.787857
```

The image below illustrates the plot.



Question 6 – Lambda Function (merit)

In the file `q6.py` write an ordinary function called `solution` which returns a list of strings obtained from the input list of strings of equal length, whereby in every string the character at the specified position is replaced with the given character. The ordinary function solution() must consist of one return statement alone, that features **either**

- the `map()` built-in function

OR

- list comprehension

OR

- generator comprehension

to apply the lambda function created in Question 1 to every element of the input list of strings.

Arguments:

- A list of strings of equal length (consisting of any number of characters)
- A string containing one character (the new character to replace an existing character in the original string)
- A non-negative integer in the range between 0 and length of the string - 1 (containing the position at which the existing character needs to be replaced with the new character).

Return value:

- A list of strings where each string has the character at the specified position replaced by the given character.

For example:

- The list ['$153.25', '$100.50', '$199.99', '$300.00'] with parameters '£' and 0 passed to the ordinary function solution() should return the list ['£153.25', £100.50', '£199.99', '£300.00']
- The list ['toilet', 'tamra'] with parameters ' ' (one space character) and 2 passed to the ordinary function solution() should return the list ['to let', 'ta ra']
- The list ['12-11-2003-17:34:54', '12-11-2003-08:14:28', '28-02-2017-12:00:00'] with parameters ' ' and 10 passed to the ordinary function solution() should return the list ['12-11-2003 17:34:54', '12-11-2003 08:14:28', '28-02-2017 12:00:00']
- An empty list returns an empty list for any character and position passed to the ordinary function solution()

Note: do not perform validation on any of the parameters. This is the responsibility of the function caller outside the ordinary function solution().

Question 7 – Object-oriented (merit)

In the file `q7.py` create the class `Trainee` that includes the following:

- the class attribute count_trainees, to store the number of trainees
- the constructor, accepting the following instance parameters:
  - trainee_id (int)
  - first_name (str)
  - last_name (str)
  - date_joined (str)
  - stream (str)
  - weeks (int)

  to initialize the above 6 and the below 3 additional instance attributes:
  - email (str)
  - date_left (str) – set to None
  - courses (empty dictionary), where key is the course name and value is the course's mark

  and to increment the class attribute count_trainees.
  The email address should be constructed from the lowercased first_name joined with the lowercased last_name by the dot (.) character in between, followed by @fdmgroup.com.
- the class method print_count() to print out the number of trainees, i.e. "The number of trainees is 1"
- the instance methods:
  - assign_course() – to add a course to the courses dictionary (with mark 0) ensuring that the number of courses in the dictionary does not exceed the number of weeks, as each course lasts for at least a week. In case no more courses can be added to the courses dictionary assign_course() displays an appropriate message
  - assign_mark_for_course(), to assign a mark to a course, checking that the course exists in the dictionary courses
  - avg_mark(), to work out the average mark for the Trainee
  - terminate_employment(), to be called for every trainee that leaves FDM. It assigns the date passed to it as argument to the instance attribute date_left and decrements the class attribute count_trainees.

The instance attributes trainee_id, email and date_left must be read only (once set they cannot be modified by an object). Any attempt to modify these attributes directly by an object should display an appropriate message. The date_left can only be changed through terminate_employment() method. The instance attributes first_name and last name should be accessed only through the getter and the setter methods. The first_name's and last name's setters must update the instance attribute email accordingly and that is the only way to set and modify the email.

Note: unit tests cannot test the correctness of methods that print out text; upon running the script q7-test, such text should be displayed in blue for you to check.

Question 8 – NumPy (merit)

In the file `q8.py` write a Python function called `hlookup()` that finds the value in the first row of a given NumPy array and returns the corresponding value in a specified row (implement the Excel HLOOKUP function with the exact match). The function returns the value '-1' (as string) if lookup value is not found in the first row, and 'None' (as string) if the given row does not exist within the data set. Note that the function always returns a string: it contains either the found value, '-1' or 'None'. The function accepts an array but the data set is given in the file data_set_1, with TAB separating the data.

Arguments:

- An array of strings consisting of values from a given data_set
- A string containing the value looked up for in the first row
- An integer representing the row index where to look for a matching value (e.g. 1 represents 1st row, 2 represents 2nd row etc.)

Return value:

- A string containing either the found value, '-1' or 'None'

For example:

- The array obtained from data_set_1, with the lookup value 'WX-534' and row number 5 should return '539'

- The array obtained from data_set_1, with the lookup value 'UW-698' and row number 8 should return '2053'
- The array obtained from data_set_1, with the lookup value 'UW-698' and row number 9 should return 'None', as there is no row 9
- The array obtained from data_set_1, with the lookup value 'XY-123' and row number 5 should return '-1', as 'XY-123' is not present in the first row

Question 9 – Pandas (merit)

In the file `q9.py` write a function called `summary_grades_2021_UK` that uses Pandas to work out the total number of exams, number of distinctions, number of merits, number of passes and number of fails in the specified academies passed to the function within a list. The criteria to work out the grade from the percentages are given in the following table:

Percentage	Grade
>= 90	Distinction
>= 80 and < 90	Merit
>= 75 and < 80	Pass
>=0 and < 75	Fail

Important: Use the `'with'` statement to safely load and safely read data from the input file. Use the provided file "existing_unreadable_file" included to the exam files to test your code. In case the input file is not found, the following message should be returned (not displayed): The file "inexistent_file" does not exist. In case the input file is found, but is not readable, the following message should be returned (not displayed): Error reading file "existing_unreadable_file".

Arguments:

- A string storing the file name containing the data set. The file is expected to be found in the same folder where the script q9.py, containing the function definition is stored
- A list of strings containing the academies for which to work out the grades

Return value:

- The DataFrame consisting of one row for each academy and 6 columns: Academy, Total No. Exams, Distinction, Merit, Pass, Fail. The intersection of any row and column contains the relevant calculated value (e.g. the intersection between 'Leeds' row and 'Pass' column contains the number of exams in Leeds that achieved 'Pass' grade.
- The string containing the message: The file "filename" does not exist, in case the input file "filename" is not found in the current directory
- The string containing the message: Error reading file "filename", in case the input file "filename" contains unreadable character(s)

For example:

- For the file 'fdm_training_data_2021.csv' and the list ['London', 'Leeds', 'Glasgow'], the function should return the following DataFrame:

```
     Academy  Total No. Exams  Distinction  Merit  Pass  Fail
0     London             6802         2991   2172   970   669
1      Leeds             3249         1184   1199   622   244
2    Glasgow             3241         1399   1190   400   252
```

- An empty list should return an empty DataFrame

Note: To pass unit testing:

1. the order and spelling of the column names in the returned DataFrame must match the order and spelling of the column names in the above example: Academy, Total No. Exams, Distinction, Merit, Pass, Fail.
2. the two messages to be displayed in case the input file is not found in the current directory, and in case the input file contains unreadable character(s), must be identical to the ones shown above in red (with "filename" replaced with the file name being tested and stated within double quotes).

Question 10 – Matplotlib (merit)

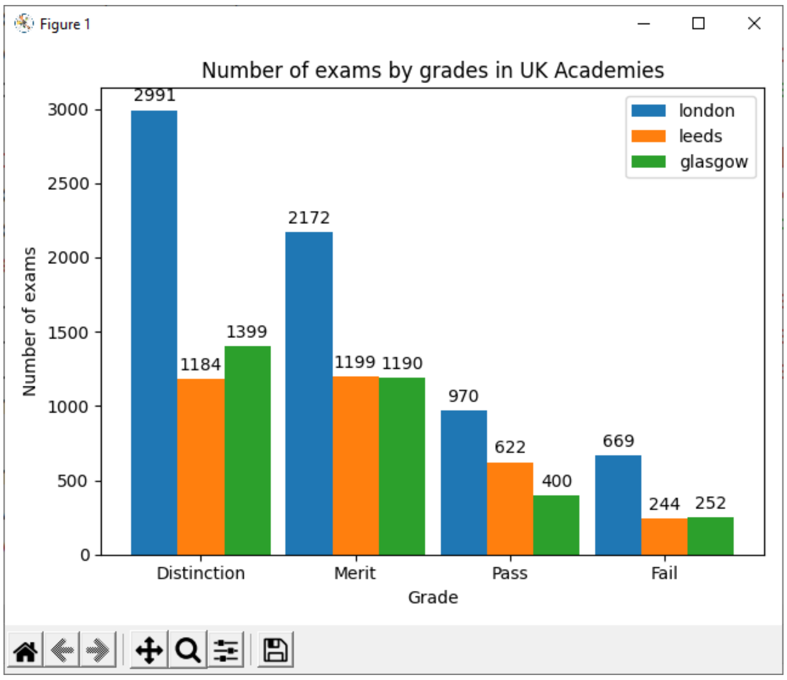
Using the data from the provided input file "fdm_training_data_2021.csv", in the file `q10.py` create a **Matplotlib bar plot** showing the number of exams for each grade in each of the three UK academies. Include values for the bars at the top of the bars.

Draw the plot in **one** of the following two ways:

1. If you have created the function `summary_grades_2021_UK()` for Question 9, import it and use it to generate the data for the plot
2. If you have not created the function `summary_grades_2021_UK()`, create the DataFrame using the data from the example given for Question 9:

```
    Academy  Total No. Exams  Distinction  Merit  Pass  Fail
0    London             6802         2991   2172   970   669
1     Leeds             3249         1184   1199   622   244
2   Glasgow             3241         1399   1190   400   252
```

The image below illustrates the plot.



Question 11 – Lambda Function (distinction)

In the file `q11.py` create an ordinary function `solution` which uses the lambda function created in Question 1 **either**

- passed as an argument to a higher order function

OR

- returned from a higher order function

to obtain a solution that applies the lambda function created in file q1 for Question 1 to every element of a list of strings.

Note: the higher order function needs to be defined separately and used (called) within the provided ordinary function called `solution`. The higher order function can be defined as an ordinary or lambda function, but the lambda function created in Question 1 needs to be either passed as an argument to the higher order function, or returned from a higher order function.

Arguments:

- A list consisting of any number of strings where each string is of equal length
- A string containing one character (the new character to replace an existing character in every string within the list)
- A non-negative integer in the range between 0 and length of the string - 1 (containing the position at which the existing character needs to be replaced with the new character).

Return value:

- A list of strings where each string has the character at the specified position replaced by the given character.

For example:

- The list [‘$153.25’, ‘$100.50’, ‘$199.99’, ‘$300.00’] with parameters ‘£‘ and 0 passed to the ordinary function solution() should return the list [‘£153.25’, £100.50’, ‘£199.99’, ‘£300.00’]
- The list [‘toilet’, 'sorrow'] with parameters ‘ ‘ (one space character) and 2 passed to the ordinary function solution() should return the list [‘to let’, 'so row']
- The list ['12-11-2003-17:34:54', '12-11-2003-08:14:28', '28-02-2017-12:00:00'] with parameters ‘ ‘ and 10 passed to the ordinary function solution() should return the list ['12-11-2003 17:34:54', '12-11-2003 08:14:28', '28-02-2017 12:00:00']
- An empty list returns an empty list for any character and position passed to the ordinary function solution()

Note: do not perform validation on any of the parameters. This is the responsibility of the function caller outside the ordinary function solution().

Question 12 – Object-oriented (distinction)

Both trainers and trainees are FDM employees. In the file `q12.py` create a class `Employee` as parent of the classes Trainer and Trainee created in Questions 2 & 7 respectively, to define a common Application Program Interface(API). The file q12.py will include Employee, Trainer and Trainee classes.
The class Employee should not be instantiable, and should include the following:

- class attributes:
  - company (read-only and set to FDM Group)
  - count_employees (storing the number of employees at FDM). The value of count_employees will always show the total of count_trainers and count_trainees.
- the constructor, to initialize the common instance attributes of Trainer and Trainee classes: employee_id (int), first_name (str), last_name (str), email (str) and date_joined (str), and to set date_left (str) to None. Define the employee_id's getter and setter as abstract methods to enforce their implementation in Trainer and Trainee classes, where their implementation can be customised accordingly (see below under changes to be made in Trainer and Trainee classes). The email address should be constructed from the lowercased first_name joined with the lowercased last_name by the dot (.) character in between, followed by @fdmgroup.com. The instance attributes employee_id, email and date_left must be read only (once set they cannot be modified by an object). An attempt to change email directly by an object should produce the message: "email can be modified only through changing the first_name or last_name of an employee". An attempt to change the date_left directly by any employee should produce the message: "date_left cannot be modified directly by any employee; date_left can be set only through terminate_employment() method" .
  The instance attributes first_name and last name should be accessed only through the getter and the setter methods. The first_name's and last name's setters must update the instance attribute email accordingly.
  Make sure the Employee's subclasses are forced to implement the constructor.
- the class methods:

- print_count() to display the number of FDM employees (if called by Employee class), the number of FDM trainers (if called by Trainer class) and the number of FDM trainees (if called by Trainee class). Use the built-in class attribute __name__ to include the type of employee to the message, such as: "The number of employees is 3", "The number of trainers is 1", or "The number of trainees is 2" and to work out their correct number. The print_count() method should be implemented only in the Employee class - it will be inherited to Employee's subclasses, and it should display the appropriate message regardless of what class it will be called from: Employee, Trainer or Trainee.
- increment_count() to increment the count_employees whenever a new trainer or a new trainee joins FDM and display the message: "New employee added."
- decrement_count() to decrement the count_employees whenever a trainer or a trainee leaves FDM and display the message: "Employee left."

Amendments to be made in Trainer class:
Amend the Trainer's constructor by replacing initializations of instance variables employee_id, first_name, last_name and date_joined with the call to Employee's constructor, and to call the Employee's method to increment the Employee's class attribute count_employees whenever a new trainer joins FDM.
The setter implementation for the employee_id in Trainer class needs to assign the employee_id value to the attribute trainer_id and display the message "Setting trainer_id to < employee_id>" in case the trainer_id attribute does not already exist, otherwise the following message should be displayed: "Attempting to alter read-only attribute: trainer_id". This will ensure that employee_id instance attribute of the Employee class is implemented in Trainer class as trainer_id and that it cannot be changed once set from the Trainer's constructor.
The getter implementation for the employee_id in Trainer class needs to return the value of the trainer_id attribute.
Amend the Trainer's terminate_employment() method to call the Employee's method to decrement the Employee class attribute count_employees whenever a trainee leaves FDM.

Remove the print_count() class method from the Trainer class (as it should now be present only in the Employee class).

Amendments to be made in Trainee class:
Amend the Trainee's constructor by replacing initializations of instance variables employee_id, first_name, last_name and date_joined with the call to Employee's constructor, and to call the Employee's method to increment the Employee class attribute count_employees whenever a new trainee joins FDM.
The setter implementation for the employee_id in Trainee class needs to assign the employee_id value to the attribute trainee_id and display the message "Setting trainee_id to <employee_id>" in case the trainee_id attribute does not already exist, otherwise the following message should be displayed: "Attempting to alter read-only attribute: trainee_id". This will ensure that employee_id instance attribute of the Employee class is implemented in Trainee class as trainee_id and that it cannot be changed once set from the Trainee's constructor.
The getter implementation for the employee_id in Trainee class needs to return the value of the trainee_id attribute.
Amend the Trainee's terminate_employment() method to call the Employee's method to decrement the Employee class attribute count_employees whenever a trainee leaves FDM.
Remove the print_count() class method from the Trainee class (as it should now be present only in the Employee class).

Note: unit tests cannot test the correctness of methods that print out text; upon running the script q12-test, such text should be displayed in blue for you to check.

Question 13 – NumPy (distinction)

A DVD rental company assembled its items' loans over 12 months of a year, stored in the flat file data_set_2. In the file `q13.py` write the function called `total_yearly_loans` that accepts the NumPy array obtained from the file data_set_2, calculates the total of loans for each month, and returns a 2-D array of 2 rows and 12 columns with the following headings: Tot. Loans Jan, Tot. Loans Feb, Tot. Loans Mar, …, Tot. Loans Dec. Below each heading, in the 2nd row the

array shows the number of loans for the month in the heading above it.
Before returning the output array, the function should save it to a text file named "tot_yearly_loans.txt", using the semicolon as delimiter.

Arguments:

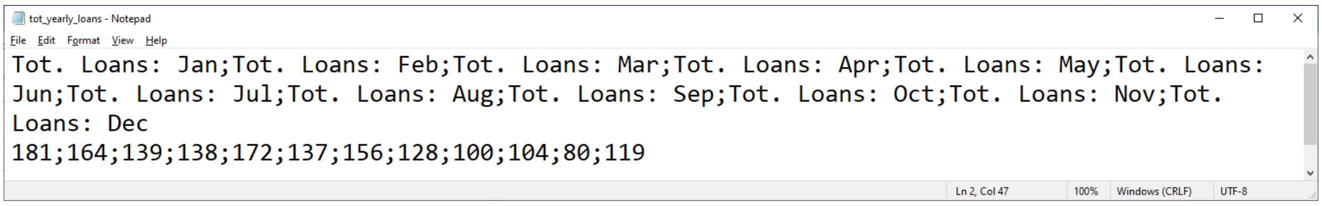
- An array of strings consisting of values from the given file data_set_2.txt

Return value:

- An array of strings with 2 rows and 12 columns where the 1st row lists the headings as stated above and the 2nd row displays total number of loans for each of the 12 months.
  The function also saves the returned array to a text file named 'tot_yearly_loans.txt', using semicolon to separate the data.

For example:

- The array obtained from data_set_2, should return the 2-D array:
  ```
  [['Tot. Loans: Jan' 'Tot. Loans: Feb' 'Tot. Loans: Mar' 'Tot. Loans: Apr'
    'Tot. Loans: May' 'Tot. Loans: Jun' 'Tot. Loans: Jul' 'Tot. Loans: Aug'
    'Tot. Loans: Sep' 'Tot. Loans: Oct' 'Tot. Loans: Nov' 'Tot. Loans: Dec']
   ['181' '164' '139' '138' '172' '137' '156' '128' '100' '104' '80' '119']]
  ```
  and create the following text file:



```
Tot. Loans: Jan;Tot. Loans: Feb;Tot. Loans: Mar;Tot. Loans: Apr;Tot. Loans: May;Tot. Loans:
Jun;Tot. Loans: Jul;Tot. Loans: Aug;Tot. Loans: Sep;Tot. Loans: Oct;Tot. Loans: Nov;Tot.
Loans: Dec
181;164;139;138;172;137;156;128;100;104;80;119
```

Question 14 – Pandas (distinction)

In the file `q14.py` write the function called `summary_grades_per_activity` to work out and show the number of fail, pass, merit and distinction grades for a specified course code and each different combination of (activity name, attempt, grade) values. The grade needs to be worked out from the percentage values using the same criteria as shown in the table in question 9. In addition to these grades, include the 'unknown' grade for cases where the percentage value is missing.

Arguments:

- A string storing the file name containing the data set. The file is expected to be found in the same folder where the script q14.py, containing the function definition is stored
- A string storing the course code for which to show the grades' summary

Return value:

- The DataFrame consisting of one row for each (course code, activity name, attempt, grade) combination within the specified course code, and 5 columns (Course Code, Activity Name, Attempt, Grade, Total).

For example:

- For the file 'fdm_training_data_2021.csv', course code 'L-21-FOU-02', the function should return the following DataFrame:

```
    Course Code                                 Activity Name  Attempt    Grade  Total
0   L-21-FOU-02             Core - Business Fundamentals Exam        1  unknown      0
1   L-21-FOU-02             Core - Business Fundamentals Exam        1        F      1
2   L-21-FOU-02             Core - Business Fundamentals Exam        1        P      1
3   L-21-FOU-02             Core - Business Fundamentals Exam        1        M      1
4   L-21-FOU-02             Core - Business Fundamentals Exam        1        D     10
5   L-21-FOU-02             Core - Business Fundamentals Exam        2  unknown      0
6   L-21-FOU-02             Core - Business Fundamentals Exam        2        F      0
7   L-21-FOU-02             Core - Business Fundamentals Exam        2        P      0
8   L-21-FOU-02             Core - Business Fundamentals Exam        2        M      0
9   L-21-FOU-02             Core - Business Fundamentals Exam        2        D      1
10  L-21-FOU-02     Core - Business Fundamentals Presentation        1  unknown      0
11  L-21-FOU-02     Core - Business Fundamentals Presentation        1        F      0
12  L-21-FOU-02     Core - Business Fundamentals Presentation        1        P      6
13  L-21-FOU-02     Core - Business Fundamentals Presentation        1        M      2
14  L-21-FOU-02     Core - Business Fundamentals Presentation        1        D      5
15  L-21-FOU-02     Core - Business Fundamentals Presentation        2  unknown      0
16  L-21-FOU-02     Core - Business Fundamentals Presentation        2        F      0
17  L-21-FOU-02     Core - Business Fundamentals Presentation        2        P      0
18  L-21-FOU-02     Core - Business Fundamentals Presentation        2        M      0
19  L-21-FOU-02     Core - Business Fundamentals Presentation        2        D      0
20  L-21-FOU-02           EMEA Pro Skills I/V Assessment 2021        1  unknown      0
21  L-21-FOU-02           EMEA Pro Skills I/V Assessment 2021        1        F      0
22  L-21-FOU-02           EMEA Pro Skills I/V Assessment 2021        1        P      2
23  L-21-FOU-02           EMEA Pro Skills I/V Assessment 2021        1        M      3
24  L-21-FOU-02           EMEA Pro Skills I/V Assessment 2021        1        D      9
25  L-21-FOU-02           EMEA Pro Skills I/V Assessment 2021        2  unknown      0
26  L-21-FOU-02           EMEA Pro Skills I/V Assessment 2021        2        F      0
27  L-21-FOU-02           EMEA Pro Skills I/V Assessment 2021        2        P      0
28  L-21-FOU-02           EMEA Pro Skills I/V Assessment 2021        2        M      0
29  L-21-FOU-02           EMEA Pro Skills I/V Assessment 2021        2        D      0
30  L-21-FOU-02  EMEA Pro Skills Presentation Assessment 2021        1  unknown      0
31  L-21-FOU-02  EMEA Pro Skills Presentation Assessment 2021        1        F      0
32  L-21-FOU-02  EMEA Pro Skills Presentation Assessment 2021        1        P      0
33  L-21-FOU-02  EMEA Pro Skills Presentation Assessment 2021        1        M      4
34  L-21-FOU-02  EMEA Pro Skills Presentation Assessment 2021        1        D     10
35  L-21-FOU-02  EMEA Pro Skills Presentation Assessment 2021        2  unknown      0
36  L-21-FOU-02  EMEA Pro Skills Presentation Assessment 2021        2        F      0
37  L-21-FOU-02  EMEA Pro Skills Presentation Assessment 2021        2        P      0
```

```
38  L-21-FOU-02  EMEA Pro Skills Presentation Assessment 2021       2        M      0
39  L-21-FOU-02  EMEA Pro Skills Presentation Assessment 2021       2        D      0
40  L-21-FOU-02       EMEA Pro Skills Written Assessment 2021       1  unknown      0
41  L-21-FOU-02       EMEA Pro Skills Written Assessment 2021       1        F      0
42  L-21-FOU-02       EMEA Pro Skills Written Assessment 2021       1        P      1
43  L-21-FOU-02       EMEA Pro Skills Written Assessment 2021       1        M      3
44  L-21-FOU-02       EMEA Pro Skills Written Assessment 2021       1        D     10
45  L-21-FOU-02       EMEA Pro Skills Written Assessment 2021       2  unknown      0
46  L-21-FOU-02       EMEA Pro Skills Written Assessment 2021       2        F      0
47  L-21-FOU-02       EMEA Pro Skills Written Assessment 2021       2        P      0
48  L-21-FOU-02       EMEA Pro Skills Written Assessment 2021       2        M      0
49  L-21-FOU-02       EMEA Pro Skills Written Assessment 2021       2        D      0
50  L-21-FOU-02                      EXCEL Exam - March 2017       1  unknown      0
51  L-21-FOU-02                      EXCEL Exam - March 2017       1        F      3
52  L-21-FOU-02                      EXCEL Exam - March 2017       1        P      1
53  L-21-FOU-02                      EXCEL Exam - March 2017       1        M      3
54  L-21-FOU-02                      EXCEL Exam - March 2017       1        D      7
55  L-21-FOU-02                      EXCEL Exam - March 2017       2  unknown      0
56  L-21-FOU-02                      EXCEL Exam - March 2017       2        F      0
57  L-21-FOU-02                      EXCEL Exam - March 2017       2        P      0
58  L-21-FOU-02                      EXCEL Exam - March 2017       2        M      1
59  L-21-FOU-02                      EXCEL Exam - March 2017       2        D      1
60  L-21-FOU-02                   EXCEL Project - March 2017       1  unknown      0
61  L-21-FOU-02                   EXCEL Project - March 2017       1        F      2
62  L-21-FOU-02                   EXCEL Project - March 2017       1        P      4
63  L-21-FOU-02                   EXCEL Project - March 2017       1        M      6
64  L-21-FOU-02                   EXCEL Project - March 2017       1        D      2
65  L-21-FOU-02                   EXCEL Project - March 2017       2  unknown      0
66  L-21-FOU-02                   EXCEL Project - March 2017       2        F      0
67  L-21-FOU-02                   EXCEL Project - March 2017       2        P      1
68  L-21-FOU-02                   EXCEL Project - March 2017       2        M      0
69  L-21-FOU-02                   EXCEL Project - March 2017       2        D      0
```

Note 1: Use the following lines of code to display the untruncated DataFrame:

```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)
```

Note 2: To pass unit testing the order and spelling of the column names in the returned DataFrame must match the order and spelling of the column names in the above example: Course Code, Activity Name, Attempt, Grade, Total. Ensuring safe loading and safe reading data from file is not required for this task.

Question 15 – Seaborn (distinction)

In the file `q15.py` import the function `summary_grades_per_activity` produced for Question 14 and use it to create a **Seaborn bar plot** showing the total number of each of the 5 grade values (unknown, fail, pass, merit, distinction)

for each activity in L-21-FOU-02 course, across all attempts (within each activity add up the number of grades for each grade value across all attempts).
For example, you can see from the given example in Question 14 that the grade breakdown for the activity "Core - Business Fundamentals Exam" is: unknown – 0, P – 1, F – 1, M - 1, D – 10 in the first exam attempt, and unknown – 0, P – 0, F – 0, M - 0, D – 1 for the second exam attempt. The plot needs to show the totals across all attempts (here two), hence: unknown – 0, P – 1, F – 1, M - 1, D – 11.

Each activity with its grades' breakdown should be displayed as a separate subplot within a multiple row plot. Include values for the bars at the top of the bars.

If you have not created the function `summary_grades_per_activity()`, create the DataFrame using the data from the example given for Question 14.

The image below illustrates the plot.

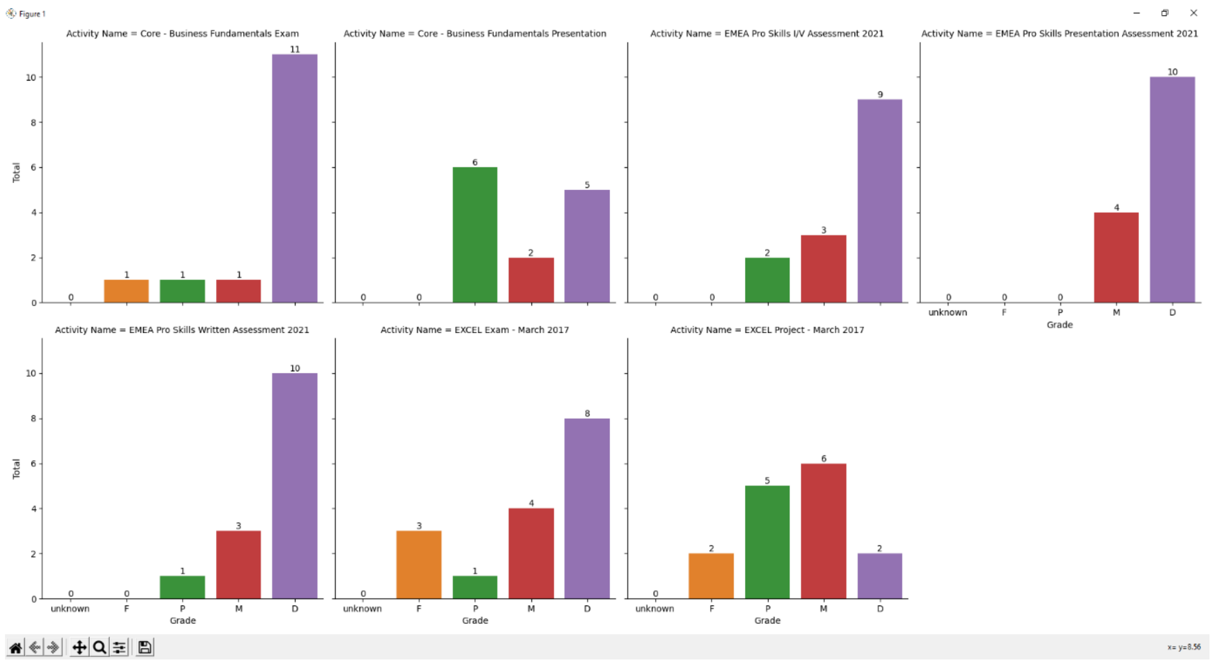


Figure 1
Activity Name = Core - Business Fundamentals Exam
Activity Name = Core - Business Fundamentals Presentation
Activity Name = EMEA Pro Skills I/V Assessment 2021
Activity Name = EMEA Pro Skills Presentation Assessment 2021
Activity Name = EMEA Pro Skills Written Assessment 2021
Activity Name = EXCEL Exam - March 2017
Activity Name = EXCEL Project - March 2017
Total
Grade
unknown F P M D
x= y=8.56