Noroff University College

Assessment Cover Sheet and Feedback Form
2019/2020

Course Code:	**Course Title:**	**Lecturer:**
UC1PR1102	Introduction to Programming	Rayne Reid Johan Van Niekerk
Assignment No: 3	**Total number of pages, including this page:** 14	**Minimum Word Count: Code & 200+ word reflection all inside the provided .ipynb file.**
Assignment Title: **Assessment 3:** Report on laboratory work		
Date Released:	**Submission Date:**	**Feedback Date:**
12.10.2020	26.10.2020 (14:00)	23.11.2020

Section 1: Submission

Record of Submission and Plagiarism Declaration

In submitting this work, you declare that this assignment is your own work and that any collaborative work or existing material has been specifically indicated in the text. In submitting this assignment, you agree that this work may be submitted for plagiarism testing.

Work should be submitted:

- Electronically via Moodle in all cases.
- If problems are encountered, you may email the course staff for assistance. However, it remains the student's responsibility to ensure the submission to the LMS was made.

DELIVERABLE

- The completed Jupyter Notebook fie, with the code full run and results visible.

If the marker must run your file when you mark it there will be an immediate 10% penalty on your final mark.

Late submissions will result in zero grade, unless agreed with course leader in advance.

IT IS YOUR RESPONSIBILITY TO KEEP A COPY OF ALL SUBMITTED WORK.

Contents

Table of Figures

Section 2: Assignment Details

This section presents the required theory for completing the assessment, the assessment's application details, and the assessment criteria.

Provided code

To complete the assessment which will be explained in the forthcoming subsections you are being provided with a few files. These will be briefly explained here and referred to in relation in later subsections, as necessary. The provided files are:

1. `Assessment%20-%20Anagrams.ipynb`

This is the file in which you will complete the required code. It has all required function definitions, and some code included with in it. You must use only these functions to complete the final solution. However, you can add additional libraries, file references, and variable related code, as necessary.

2. `words.txt`

This is the full dictionary file to be used within the assignment. It is the same dictionary used in the solution from which the screenshots were taken. The dictionary does not contain the words "I" or "A". It is important you include these words in the dictionary or dictionary list before accessing the list of dictionary words in the activity functions.

3. `smaller.txt`

This is a smaller version of the dictionary which you have the option of using to test functionality of the activities. You do not need to use it if you do not want to. It is intended to make testing your code quicker, as some activities, particularly **Activity B** can take a longer amount of time to run using the ***words.txt*** file. **When you submit the final solution make sure your solution is using the words.txt**

4. `racecar-10.09.2020_13.05.06.txt`

This is an example of a file which would be output by **Activity A**. This example shows the expected file output when there are no anagrams for the user provided word, but the provided word itself is a palindrome.

5. `assessment-10.09.2020_13.05.00.txt`

This is an example of a file which would be output by **Activity A**. This example shows the expected file output when there are no anagrams for the user provided word, but the provided word itself is not a palindrome.

6. `repaper-10.09.2020_13.05.03.txt`

This is an example of a file which would be output by **Activity A**. This example shows the expected file output when there are anagrams for the user provided word, and the provided word itself is a palindrome.

7. `dictionaryPalingrams.txt`

This is an example of a file which would be output by **Activity C**. It is an incomplete file.

Prerequisite knowledge for assessment
The assessment will require you to find anagrams, palindromes, and palingrams. To do this you need to understand what they are. The below section presents a simple explanation of these concepts.

1. Anagrams

An *anagram* is a play on words created by rearranging the letters of the original word to make a new word or phrase. Anagram examples can be fun and witty, and they often end in hilarious results.

Anagrams are often longer words or phrases that do not necessarily mean anything but are fun to come up with and say. There are also anagrams of simple words that create random, new words that are not relevant to one another.

Single word anagram examples include:

- Tar = Rat
- Arc = Car
- Elbow = Below
- State = Taste
- Cider = Cried
- Dusty = Study
- Night = Thing

There are more complex phrases which include multiple words; however, this assessment will focus on anagrams of single words found in the dictionary.

2. Palindromes
A *palindrome* is a word, phrase, number, or sequence of words that reads the same backward as forward. Punctuation and spaces between the words or lettering is allowed. This assessment will focus on single word, and two-word palindromes.

Single word palindrome examples include:

- anna
- civic
- kayak
- level

- madam
- mom
- noon
- racecar
- radar
- redder

3. Palingrams

A *palingram* is a sentence in which the letters, syllables, or words read the same backward as they do forward.

Multiple word palindromes (palingrams) include:

- top spot
- my gym
- ala cicala

Overview

This assessment requires you to create an application, which can perform three activities. The exact details of each activity will be explained in their own sub-sections. This section will outline the primary high-level logic of the program – the main function logic.

This application will start by presenting a brief explanation of:

- anagrams, palingrams and palindromes (Prerequisite knowledge), and
- the three activities which can performed by the application (Activity A, Activity B, Activity C).

The function with this initial text is included in the function called `print_explanation()`. The application should call this function to show the display in figure 1.

```
An anagram is a word or phrase formed by rearranging the letters of a different word or phrase,
typically using all the original letters exactly once. It can however us on a few of the letters.
A palindrome is a word, number, phrase, or other sequence of characters which reads the same
backward as forward, such as "madam" or "racecar".

A palingram – a letter palingram. A palingram can also be created using syllables
or words, in addition to individual letters. An example of this using words is, “He was, was he?”
Notice that the words can be reversed and the sentence still reads the same.
A sentence that is a palingram and a palindrome is, “I did, did I”

This application allows three activities.

  Activity A: Finds the anagrams for any word you enter. Then identifies palindromes, if there are any,
              in the identified anagrams.

  Activity B: Finds all the word pairs in the dictionary list which form palingrams.
              The palingrams ignore spaces and hyphens.

  Activity C: Find all the palindromes in the dictionary.
```

Figure 1 Display of the explanations

After the explanation is displayed the application should prompt the user to select which activity, they would like performed, or "#" to be entered to end the application (Figure 2).

```
Choose activity A,B, or C.  To run the process enter the appropriate letter. Else enter # to exit the application.
Enter choice:
```

Figure 2 Display of the prompt for activity choice or application end

The application should accept the user's choice and perform the activities processes. The processes should call the appropriate functions within the appropriate logic structures i.e. if **activity A** is selected, **only Activity A** should be run. After the Activity **A**, **B**, or **C** have run the program should always display the explanation and re-prompt the user for a new activity selection.

After the Activity **A**, **B**, or **C** have run the program should always display the explanation and re-prompt the user for a new activity selection. The application should only accept

UPPERCASE inputs of "A", "B", or "C". The only other acceptable input is "#". For any other user input the program should redisplay the explanation and re-prompt the user for the user's selection of the activity.

The application should only stop running when the user enters the "#". Otherwise it must always continue to run.

The `main()` function should contain the primary display and prompting logic.

Activity A

Figure 3 shows acceptable user input to trigger activity "A". Activity A finds the anagrams for any word a user enters.

```
Choose activity A,B, or C.  To run the process enter the appropriate letter. Else enter # to exit the application.
Enter choice:A
```

Figure 3 Acceptable input for triggering Activity A

The program prompts the user for their choice of a single word.

The application must check that the user's input is a word or alphabetical characters. It can not accept special characters or numbers. If the input is invalid the initial explanation and user prompting should be re-displayed. The application should not need to be restarted manually.

If the user was correct, the application should display the message "`Using word =`" with the user's input word added at the end. See figure 4, figure 5, and figure 6.

Once the sentence is displayed the application should use a combination of logic and the provided functions to find the anagrams of the provided words. The anagrams should be found by taking the user a word and searching the dictionary for anagrams of equal length for the word. The application should compile a list of the anagrams if they exist. It should then print the list of anagrams

- The first item in the anagram list should always be the user's word
- All other items in the list should be sorted alphabetically (see figure 6)

Once the anagrams list has been created, all the items in the list should be printed as shown in figure 4. The displaying of the list should be accomplished using the `print_generic_list()` function. This same function should include code to display a descriptive message of there being no anagrams to display of the list contains no items. This code should not change if the function was called elsewhere and provided different parameters.

After displaying the anagrams, the application should search the anagrams list for any palindromes. The application should compile a list of the palindromes if they exist in the list of anagrams. The list should include the word itself if it is a palindrome. The final part of this functionality should display the message:

"Number of palindromes found = " and the number of palindromes in the list. e.g. "Number of palindromes found = 12"

Once the list of palindromes is formed its contents should be displayed using the `print_generic_list()` function. If there are no palindromes the appropriate message should be displayed (see figure 5).

After all the processing and the displaying of the information the entire result set should be written to file. The `write_anagrams_palindromes_to_file()` function should be used to accomplish the task.

The function should store the list of anagrams, and the list of palindromes. The format of how this is done is shown in the included example text examples.

1. **racecar-10.09.2020_13.05.06.txt**
2. **assessment-10.09.2020_13.05.00.txt**
3. **repaper-10.09.2020_13.05.03.txt**

The function should create a new file for each search and process of activity A. The name should be formed using the following format:

"User_word-month.day.year_hour.minute.second.txt"

The date and time used are from the time the file is created.

```
Enter choice:A
Enter a word to be used in the anagram search: racecar
Using word = racecar

Anagrams
racecar

Number of palindromes found = 1


Palindromes
racecar
```

Figure 4 Full Activity A result display 1. Discussed in given code section.

```
Enter choice:A
Enter a word to be used in the anagram search: assessment
Using word = assessment

Anagrams
assessment

Number of palindromes found = 0


There are no Palindromes to be displayed.
```

Figure 5 Full Activity A result display 2. Discussed in given code section.

```
Enter choice:A
Enter a word to be used in the anagram search: repaper
Using word = repaper

Anagrams
repaper
paperer
prepare

Number of palindromes found = 1


Palindromes
repaper
```

Figure 6 Full Activity A result display 3. Discussed in given code section.

Activity B

Figure 7 shows acceptable user input to trigger activity "B". Activity B finds all the word pairs in the dictionary list which form palingrams. The palingrams ignore spaces and hyphens.

```
Choose activity A,B, or C.  To run the process enter the appropriate letter. Else enter # to exit the application.
Enter choice:B
Searching...
```

Figure 7 Acceptable input for triggering Activity B

Activity B uses the function `find_palingrams()`. For every word in the dictionary list, the function should check if any other word(s) in the dictionary can be used to form palingrams. Examples of palingrams are shown in the displays of figure 8 and figure 9. This function should compile a dictionary collection of the word pairs.

Once the collection of palingrams have been formed, the list should be displayed using the `print_pairs_of_twos()` functions. The display should be as shown in figure 8 and figure 9.

There are 5960-word pairings (palingrams), when using the words.txt file including the words "a" and "I". Your code should be testable using this file or any other file with a similar format.

The full collection should be written to file using the `write_palingrams_to_file()` function. The file should be written or overwrite the `file dictionaryPalingrams.txt`.

```
Enter choice:B
Searching...

Number of palingrams = 5960

a aa
a ba
a baba
a ceca
a coca
a da
a dada
a fa
a gaga
a giga
a ha
a ilia
a inia
a ixia
a ka
a kabaka
a kaka
a la
a lunula
a ma
a mama
```

Figure 8 Display of results from Activity B start of display

```
yelk ley
yell alley
yell ley
yelp ley
yet stey
yod oy
yodeled oy
yodelled oy
yoga goy
yogee goy
yogh goy
yogi goy
yogini goy
yom oy
yomim oy
yon oy
you buoy
you oy
yow oy
yuga guy
zee jeez
```

Figure 9 Display of results from Activity B end of display

Activity C

Figure 10 shows acceptable user input to trigger activity "C". Activity C finds all the palindromes in the dictionary. A sample selection of the palindromes is shown in figure 10.

```
Choose activity A,B, or C.  To run the process enter the appropriate letter. Else enter # to exit the application.
Enter choice:C

Number of palindromes found = 91


Dictionary Palindromes
aa
aba
aga
aha
ala
alula
ama
ana
anna
ava
awa
bib
bob
boob
bub
civic
dad
```

Figure 10 Sample of the display from Activity C

The application searches the entire dictionary file/list for palindromes and compiles a list of palindromes. There are 91 palindromes in the file/list of dictionary words when using the words.txt file including the words "a" and I".

The application should display all the palindromes (as seen in figure 10). This should be accomplished using the `print_generic_list()` function.

Reflective Activity

The activity of finding and manipulating anagrams, palindromes, and palingrams will not be a frequent activity in your daily professional tasks. However, the skills learned to achieve this, manipulating and find data are very transferable to working with large datasets. All the study streams will need to be able to access, manipulate, analyses and search large data files e.g. event logs, data feeds e.g. large twitter datasets, hidden file directories or properties.

Having now completed this task, you have identified the knowledge areas and skills you needed. You will also have identified which skills you are strong and weak in. In the below markdown tab, reflect on the following:

1. Which skills were you strongest in?
2. Which skills are you weakest in?
3. How do you think you can improve these skills?
4. In detail, how do you think the course's learning material could be improved to further support you and your skill development?
5. Was there anything you very strongly liked or disliked in the course?

The reflection should be a minimum of 200 words. You can reflect in a single discussion or under each heading. The headers do NOT count towards the 200-word minimum.

NOTE: This is an individual assignment. You are therefore required to work independently on your solutions. Group work will not be accepted.

Section 3: Marking and Assessment

Contribution to the final course mark:	100%
Main logic and code professionalism	25%
Activity A	30%
Activity B	20%
Activity C	15%
Reflective report	10%
Estimated number of hours to be spent on this assignment:	40

The pass grade for this assignment is 40%

IMPORTANT: If you do not submit the reflective report you immediately receive zero.

Feedback for this assessment and a breakdown of the final grade will be provided via the LMS (Moodle).

Marking Criteria Guide	
Criteria	**Functionalities relevant to criteria**
Main logic and code professionalism (25 marks)	1. Overall programming logic as per the assessment cover sheet 2. Error handling 3. Code is commented with explanations 4. The function and variables names are appropriate 5. Fully functional program
Activity A (30 marks)	1. Prompts the user for a single word 2. Check whether it's a legal entry 3. Display the word in the specified format 4. Usage of `find_anagram ()` function 5. Compile a list of anagrams if they exist and print the items in the list 6. Search the anagram list for any palindromes and compile their list if they exist 7. Usage of `print_generic_list()` to display the contents of palindromes list 8. Usage of `write_anagrams_palindromes_to_file()` to store the entire results of the activity 9. Usage of the specified convention for naming the files containing anagrams and the list of palindromes
Activity B (20 marks)	1. Usage of the function `find_palingrams ( )` to check if any other word(s) in the dictionary can be used to form palingrams 2. Display the collection of palingrams list using `print_pairs_of_twos ( )` function 3. The full collection of palingrams are written to the appropriate file using the `write_palingrams_to_file ( )` function
Activity C (15 marks)	1. Search the entire dictionary file/list for palindromes and compile a list of palindromes 2. Usage of `print_generic_list()` function to display all the palindromes
Reflective Report (10 marks)	1. Given Prompt

Mark Symbols:

F (<40)	E (40 – 49)	D (50 -59)	C (60-69)	C (70-79)	B (80-89)	A (90-100)
Fail	Sufficient	Satisfactory	Satisfactory-Good	Good	Very Good	Excellent