

1 Introduction

In this assignment you will demonstrate your knowledge and skill with the material in the unit by developing a solution to a set of problems using the concepts and techniques covered in the unit.

Your submission should include a clear description of the problems and a solution that addresses the requirements.

This assignment

Please see

2 Task

Choose one

2.1 Re

It's time to build a chat application using the concepts covered in the unit. You will be building a chat application with an intensive GUI, as reChat.

Your task is to implement the logic for the chat application. You will be given the state of a user's connection and the state of the chat application.

```
def reChatParseCommand(message, state):
    # your code here
    return action, state
```

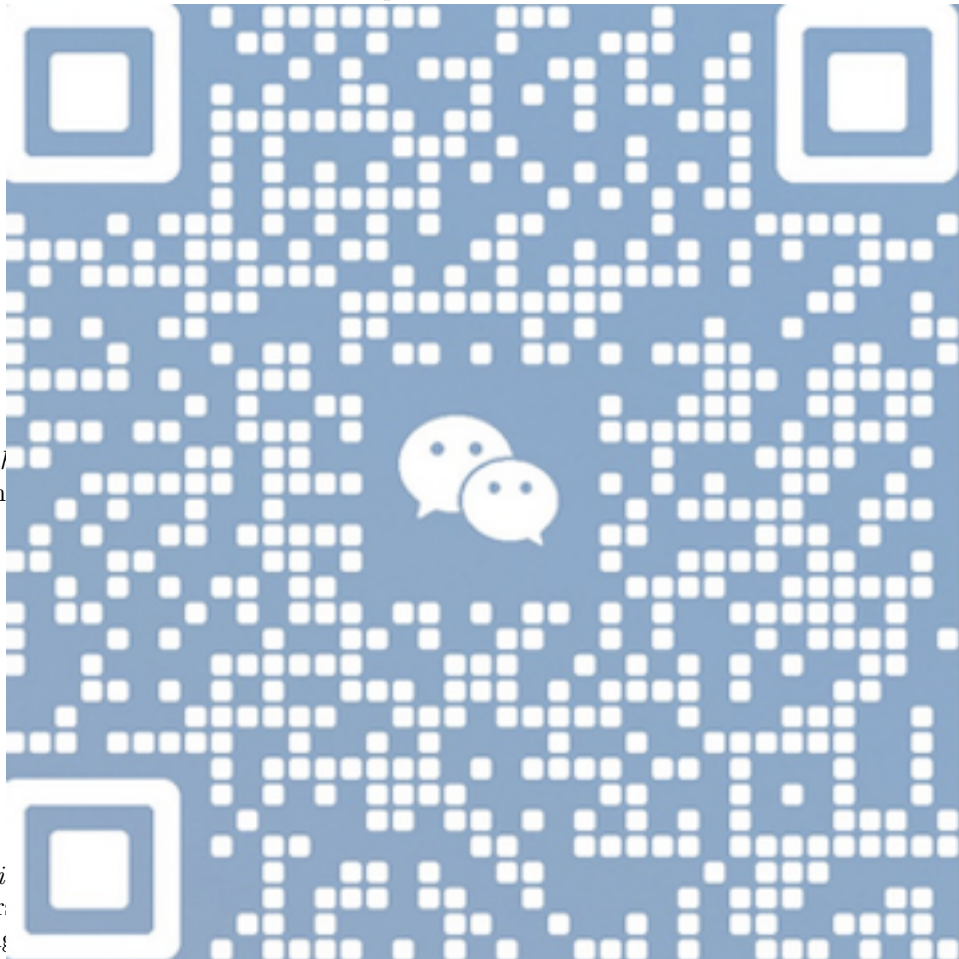
Here **message** contains the input (i.e. command or message) from the user. **state** is a variable that you can use to store whatever you like. It is initially set to **None** on the first invocation of your function. Further explanation appears below.

The returned variable **action** is a dictionary indicating what action should be taken along with additional parameters for that action, as described below. The returned variable **state** will be used as the **state**

argument on the next time the function is invoked. This mechanism allows you to keep track of the state of the connection without resorting to global variables or other mechanisms. The **state** return value is ignored by the tests.

The required behaviour is given below. Note that text surrounded by '<>' like '<spec>' in the action should be replaced with an appropriate value (eg. from the command):

- When the user first connects (indicated by the **message** argument set to '' and the **state** argument set to **None**), the action is { 'action': 'greeting' } and the connection enters *command* mode.
- In *command* mode the user has four possible commands:



ns in command

connection enters

h enters *direct*

> refers to the

the action is

'<message>',

g, <username>

the user that is

- In *cl*
chan
- In *di*
refer
being

- \leave: The connection enters *command* mode and the action is
{ 'action': 'leaveDM', 'user': '<username>' }
- \read: The connection remains in *direct message* mode and the action is
{ 'action': 'readDM', 'user': '<username>' }
- <message> (not starting with \) : The connection remains in *direct message* mode and the action is
{ 'action': 'postDM', 'user': '<username>', 'message': '<message>',
'mentions': { '<username1>', '<username2>', ...' } }

where the **mentions** value gives all the usernames that appear in <message>.

- In all other circumstances the mode does not change and the action is

```
{ 'error': 'Invalid command' }
```

The following specifications apply to channels and usernames:

- A username is @ followed immediately by a valid email address according to the limited criteria from Tutorial 10, Section 2, Question 5b. (You can use the regular expression in the tutorial solutions; there is no need for you to recreate it yourself.)
- A channel name is # followed immediately by a sequence of letters (upper or lower case) and numbers, beginning with a letter.

Please note

- Your arguments must be in the state system.
- While the connection is that you have the connection
- Your behaviour described in use.

For this task, the sample test cases in the test suite while ensuring that your solution is correct. There are 10 test cases while loops. As with the C++ solution is only worth 1 mark.

2.2 Linear

Dana is a wheat farmer. Each field she has a different rate grain bins. Each field has a different bin is different. She wants to maximise her profit by blending wheat from the bins to maximise the amount of High Protein Grade wheat she has to sell.

High Protein grade wheat needs to meet two criteria ¹:

- minimum 14% protein by weight
- maximum 12.5% moisture by weight

¹Australia currently has 32 grades for wheat determined by 47 criteria (See Wheat Standards 2023/24 All Grades). We're simplifying here.

Your task is to determine how Dana should blend the wheat from her three bins to maximise the amount of High Protein grade wheat she can sell.

Important! Solving the problem as stated in full generality goes beyond what is learned the unit (it is what's called a *linear programming* problem) so we'll restrict the problem to cases where it is possible to *exactly* match the requirements, i.e. exactly 14% protein, and exactly 12.5% moisture. This will always be possible for our test cases and there is no need for your solution to work with other cases.

Create a Python function which calculates how much wheat to use from each bin to blend together:

```
def blend(
    # you
    return
```

Your input
so:

```
Bin,Weight,Protein,Moisture
A,12,15,12.5
B,15,13.5,14
C,7,12,14
```

where the
are percent

Your function
from each
tonnes. The

```
( { 'A':
```

In order to
cases will :

Hint: You

bin required to make exactly 1 tonne of High protein grade wheat with the correct percentages for protein and moisture. This is a system of 3 equations with 3 unknowns. The second step is to scale this so that you run out of wheat in one bin.

For this task the Python code will be marked automatically according to test cases similar to the sample test cases in `test_STA_linalg.py`. You can run the sample test cases with `python test_STA_linalg.py` while ensuring that your solution is called `specialtopics.py` and is in the same directory as `test_STA_linalg.py`. There are 10 test cases in total, including one test case which checks to see if you have used any `for` or `while` loops. As with the Graphs project, your priority should be to have working code as the loops criterion is only worth 1 mark.

2.3 Probability

Recently you have inherited your long lost uncle's wheat farm, and you are determined to make a go of it. It's exciting, but there's a lot to learn. While spending some time around the local coffee shop an old farmer offers you some advice:

"You'll be wanting to get some kind of crop insurance. Input costs are high these days, so don't want to lose it all if you have crop failure. There's a few kinds of crop insurance. The expensive kind pays out for any kind of crop failure. Then there's cheaper kinds that don't pay out as well, and some that don't cover all kinds of crop failure. Plus there's some that only work for certain things. Hail insurance is like that; only pays out if your crop gets hailed out."

You ask th

"Depends
Clay tends
so some pl
grasshopper
you've got
of his, and
from that.

From Char
field: drou
recorded w

Unfortuna
else, so yo



re is a drought.
weather patterns
some places get
w seems to me
this little book
prone to what

about for your
open!). Charlie
information.

l.

not to anybody

You have managed to sign a contract with a company that will buy your entire crop at a fixed price, unless your crop fails².

Asking around to crop insurance companies, you find that there are five kinds available³:

- *Comprehensive*. Pays 80% of your contract price in the event of any kind of crop failure.
- *Hail*. Pays 80% of your contract price in the event of crop failure due to hail.

²This is not the way contracts usually work, but we're simplifying for this exercise.

³This is also not how crop insurance usually works.