Assignment 4

Key information (Front page)

How will I be assessed

A4 has a different marking schema than A2 and A3. Make sure to read everything.

Team mark (common)

This is assessed based on the team submission.

- Approximately 20% of the team mark from test cases.
- Approximately 80% of the team mark from review by your teacher as per rubric.

5% bonus marks for FIT1045 students completing the FIT1053 component

This means that the team mark can be made up for (in part) by up to 5% if you successfully complete the bonus component. For example, if you were to obtain 97% for the team mark, but have correctly solved the extra component, then your team mark becomes100%. If it was 80%, you'd get up to 85%.

Interview (individual)

The interview will happen similarly as in A2 and A3, but the individual part will have a different impact on your final mark.

After your individual interview with your tutor, you will be assigned one of four colours:

- **Green** if you can demonstrate that you understand all aspects of the work your team has submitted. You would be able to re-write the program from scratch with ease and without needing much trial and error or research.
- **Blue** if you can demonstrate that you understand most of the aspects of the work, but not all. You would be able to re-write the program from scratch with difficulty and needing some amount of trial and error and research.
- **Orange** if you can demonstrate that you have understood some aspects of the work, but master no single part of the work. You would be able to write the program from scratch, but it would take you as much effort as if you had not worked on this assignment.
- **Red** if you cannot demonstrate that you have understood any aspect of the work. You would not be able to write the program from scratch with your current understanding of the assignment and of programming.

Note that these colours will be assigned to each of you independently:

- All students of a team could get Green, if you have all contributed and ensured that you all fully understood what others have done.

- All students of a team could get Blue or less, if you have divided the work between each other and not looked into what others have done.

Team contribution factor (individual)

This is assessed based on the colour you received for the interview and your team feedback via FeedbackFruits.

The contribution factor is a real number between 0 and 1.10, reflecting your contribution to the team submission. A higher contribution to the team effort results in a high contribution factor.

Overall mark (individual)

Your overall mark for the assignment will be the product of your team mark by your contribution factor, capped at 100%.

For example,

- If your team obtains 90% of the team marks, and your contribution factor is 1.10, then your overall mark is 99%.
- If your team obtains 80% of the team marks, and your contribution factor is .5, then your overall mark is 40%.
- If your contribution factor is 0, then your overall mark is 0.

Teams smaller than 4

Some of you ended up in teams smaller than 4.

- As usual we will have a google sheet listing teams and team members. If you believe it is not up to date, please let us know via a private post on ed.
- If you are in a team of 3, we expect the same as for teams of 4.
- If your team has 2 or fewer team members, then please reach out to us so we can try to put you with another team of 2. Otherwise, as a team of 2 we expect you to complete the tasks 1, 2, and 3, but not 4.

Effective team size

After the interview, we compute the *effective size* of your team as the number of students who have received Green or Blue at the interview. If the effective size is 2 or less, but your actual team size if 3 or 4, then we will apply a correction to the team submission mark to reflect this fact, tentatively for an effect similar to the removal of the last task for teams of actual size 2.

Note that the feedback you submit for your peers is not considered to compute the above. The correction to the submission mark is only based on the interview. In other words, you would not get

a higher mark if you gave your teammates worse feedback than they deserve.

Do not try to game this mechanic by, for instance, trying to make your teammates fail the interview to get a higher mark. Such attempts will be reported as general misconduct.

Issues within your team

If you have issues within your team, such as team mates who are not responsive, you may reach out to your Pierre (AU) or Hui Xuan (MA), but you need to do so by Tuesday 17/05. After this date we will not be able to help with team issues and you will have to rely only on the team contribution factors to get a fair outcome.

Academic integrity

How will this be checked?

Your code will be compared against every other students' work in the unit.

You should also assume that anything you are able to google can be easily found by the teaching team and compared against your work.

You will also be **individually** interviewed on your understanding of different components of the program, whatever your contribution to a particular component might be. You are expected to understand all parts of the program equally well. Make sure you work as a team to ensure that.

How can I avoid academic integrity issues?

- Never copy code from anywhere. If you learn something useful online, rewrite it from scratch. It's also the best way to make sure you have understood it. If you're concerned you may cause an academic integrity case by copying something on the internet, the easiest way to avoid this is to not search anything too specific. Once you read a solution, it is very hard to forget it.
- If a fellow student asks you for a solution to a question, try instead to figure out what it is that they do not understand about the unit's content that prevents them from finding the solution themselves. Give a man a fish, and you feed him for a day. Teach a man to fish, and you feed him for a lifetime. Also, remember that giving your solution is just as much of an Academic Integrity breach as receiving it!
- You may find yourself in a situation where you feel like you physically cannot submit the assignment on time. Remember that you can submit an extension request (see Moodle AU or Moodle MA), and you can seek help (see Moodle AU or Moodle MA). If nothing works, remember that failures are part of the learning journey. And what is more important to you, an assignment mark or acting honourably?

Workspace and automated testing

We have included some test files for each of the Python files we expect you to produce (attached in this slide below).

To run these tests yourself you will need to go to the console and run

```
python test_XXX.py
```

The results you see in the console will be the results of running these tests

We will use the test cases for the 20% of your marks coming from automated tests; do not modify these files as they will simply be overwritten by your TA when it comes to marking your work.

your tutor may run similar tests with different values to what is provided in the workspace scaffold

Preparing your workspace

In this assignment, you are expected to work in teams of up to 4. You will need to meet regularly with your team and collaborate using the workspace in Ed. You should create your own workspace from scratch and upload the test files provided.

Make sure to share your workspace with you teammates. We recommend you use a single workspace, but you can use more if that works better for your team.

Once you know the tutor who will mark you, add them to your workspace prior to the interview. They will be able to see who has contributed over the course of the assessment period through the replay option.

After you submit the assignment, you will be asked to complete a peer review of one another's contributions to the team, the submission of which will contribute to your individual mark.

Test files

FIT1045

There are 5 files that correspond to chunks of work, not necessarily classes. For example, `test_character_class.py` does not test the method `make_check` of the class `Character` as we expect you to first write the class `Character`, ensure it is correct, and then write the method `make_check`.

FIT1053 / bonus marks for FIT1045

There is an extra file `test_story_chosenchar.py` corresponding to the advanced component.

Spotting differences in outputs

In case of an error, the unit test will usually show the output of you program vs the expected one. Some differences involving white space cannot be spotted in the console. If you have trouble spotting differences, we advise you to use a diff tool such as https://editor.mergely.com/.

Checking that you have the right test files and that they are not modified

One way to check that you are using the right test files is to type the command `md5sum test_*` in the terminal of your workspace. If you have the correct test files, you should see exactly this:

```
1fcdd0dc32fced0b83549efca1a25e7f  test_character_class.py
2ab69f90d03fcae87d8b80519bfe27ae  test_choose_your_own_adventure.py
6a32bc38e3e330d42283ad58b16a25a6  test_make_check_function.py
ef871298c7b40cfa3076ba1d0f586867  test_story_bestchar.py
09f9a768d1489160ac9f100163a17755  test_story_chosenchar.py
8232dc37997c2f52828342591c4213c8  test_story_class.py
```

Rubric

Submission information

When you have completed your work, download it from your workspace and upload it on the Moodle "Assignment 4 submission" box on the assessment page (Moodle AU - Moodle MA). One submission per team!

The date of submission on Moodle is used to determine late penalties.

Important: Dos and Don'ts

Do

- ✓ Add your teammates to your ed workspace.
- ✓ Run your work regularly and test with different scenarios.
- ✓ Download backup archives regularly.
- ✓ Use the class, function names and arguments given when writing your code.
- ✓ Program, review and discuss your code with your teammates.

Don’t

- ✗ Change the test files provided. We will use our own.
- ✗ Hard code expected test results. The tests your tutors run may be different to those given.
- ✗ Work or ask other members of your team to work in isolation.

Scaffold files v3

The archive contains the test files and sample data files.

 FIT1045_A4_v3.zip

v3 corresponds to change 12.

Changelog (14 changes)

Change 14 - 18/05/2022 at 14:37

In the slide on stories, added the sentence

> Two scenes might be separated by different amounts of newlines (in between two `----`).

Thanks Shien for reporting this.

Change 13 - 18/05/2022 at 14:15

Task 5 only. Updated the description to match the test cases. In the description, the arguments were described as lists of indices. They are now lists of Character objects, to match the test cases (so the test cases are unchanged). Note that this means a reversal of Change 11. Thanks to Anton and Shien for reporting this problem.

Change 12 - 17/05/2022 at 11:02

Task 5 only. Updated `test_story_chosenchar.py` which was not doing anything due to a missing main block. This is now fixed. You should download the new files if you attempt Task 5.

Md5sum updated as well here.

Thanks Shien for reporting this issue.

Change 11 - 17/05/2022 at 11:02

In task 5, edited the sample provided

```
>>> S = Story(scene_string_data,char_data)
>>> #assume C1, C2, C3 hold Max Power, Charlee Gryl and Peppa San de wej respectively
>>> S.select_character_for_check("language", [], []).get_name()
"Max Power"
>>> S.select_character_for_check("investigation", [C1,C3], []).get_name()
"Peppa San de wej"
>>> S.select_character_for_check("diplomacy", [], [C3]).get_name()
"Peppa San de wej"
>>> S.select_character_for_check("medicine", [C1,C2,C3], [C2]).get_name()
"Charlee Gryl"
```

into

```
>>> S = Story(scene_string_data,char_data)
>>> #assume C1, C2, C3 hold Max Power, Charlee Gryl and Peppa San de wej respectively
>>> S.select_character_for_check("language", [], []).get_name()
"Max Power"
>>> S.select_character_for_check("investigation", [1,3], []).get_name()
"Peppa San de wej"
>>> S.select_character_for_check("diplomacy", [], [3]).get_name()
"Peppa San de wej"
>>> S.select_character_for_check("medicine", [1,2,3], [2]).get_name()
"Charlee Gryl"
```

to match the specifications of the function. Thanks Anton for pointing that out.

Change 10 - 16/05/2022 at 11:03

Updated `test_choose_your_own_adventure.py` to use `story_class_structures` instead of `story_class_structures_bestchar` in the scaffold files. You should download the new files and replace your current copy of `test_choose_your_own_adventure.py` by the new one.

Md5sum updated as well here.

Thanks to Minuque for reporting this problem.

Change 9 - 16/05/2022 at 10:31

In this slide, under **`StoryBest.select_character_for_check`** , edited the sample input / output by changing

> >>> S = Story(scene_string_data,char_data)

into

> >>> S = StoryBest("",char_data)

as we are testing the class StoryBest and as there is no need for scene data. Thanks to Yu for reporting this.

Change 8 - 16/05/2022 at 10:17

In this slide, under "Story class string format", changed the occurrences of "SCENES:" into "SCENE" and "CHARACTERS:" into "CHARACTER", to match other places in the description and to match the test cases. Thanks Terry & Anton for pointing this out here.

Change 7 - 13/05/2022 at 22:03

Added specifications for teams of 2.

Change 6 - 13/05/2022 at 11:48

Under "Story class string format/string presentation for a single scene" in this slide, clarified that "The

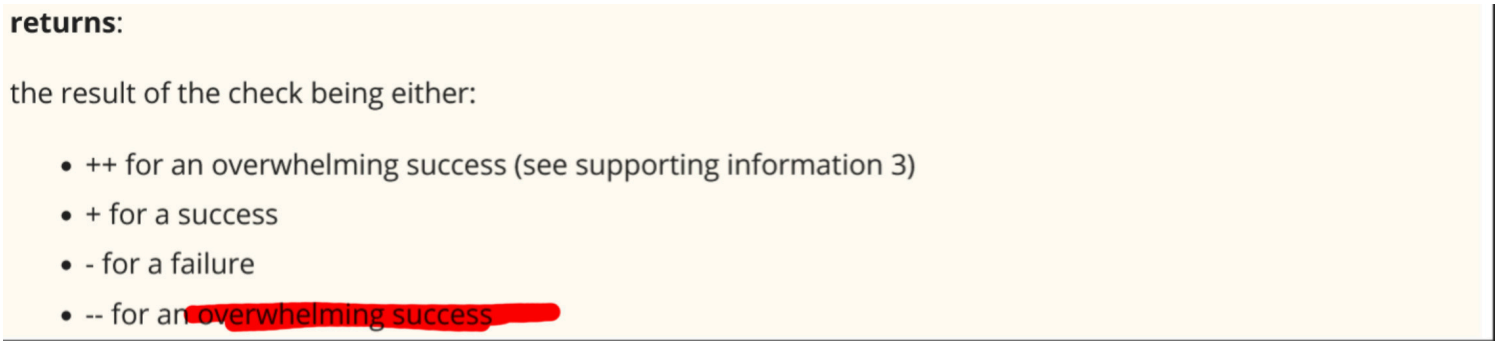
outcomes of an option should be displayed in the order ++, +, -, --". Thanks Ahmed for pointing out that this wasn't clear.

Change 5 - 12/05/2022 at 14:45

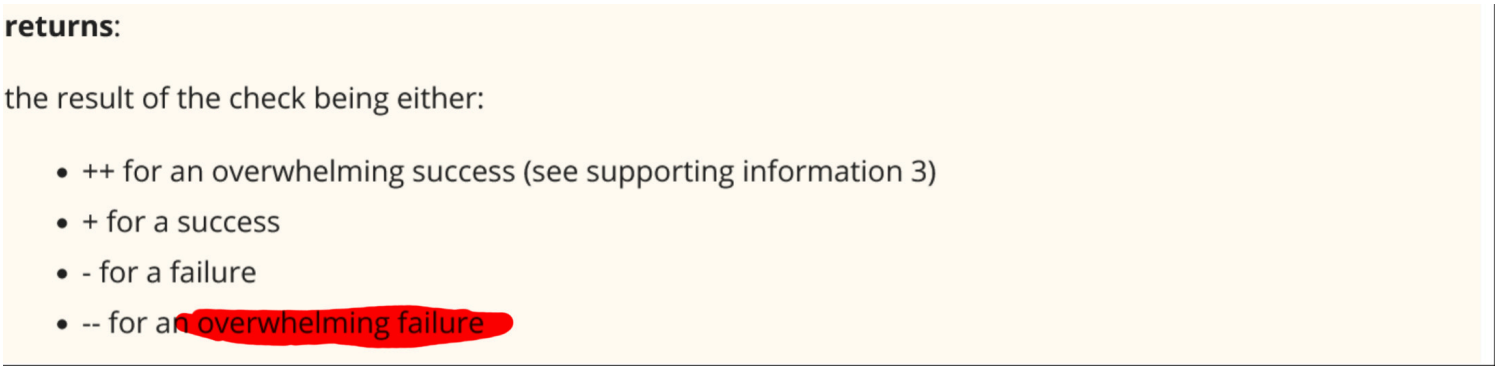
Rubric available.

Change 4 - 12/05/2022 at 10:45

Edited the description of `Character.make_check` on this slide to correct



into



Thanks to Forbes for pointing it out here.

Change 3 - 11/05/2022 at 10:15

Edited the Character Class slide to fix the discrepancy between the description and the test file. The output should be "does not equal", not "could not equal". Thanks to Muhammad for pointing this out.

Change 2 - 11/05/2022 at 10:08

Edited the File Format section of this slide to specify that the format of the character data does not change. Thanks to Xiaoyang for the question.

Change 1 - 10/05/2022 at 13:55

Edited the second paragraph of the slide "influencing an outcome" to make it a clear that the skill proficiency should be taken into account. Thank you Karleen for your questions.

Assignment overview

Overview of Assignment

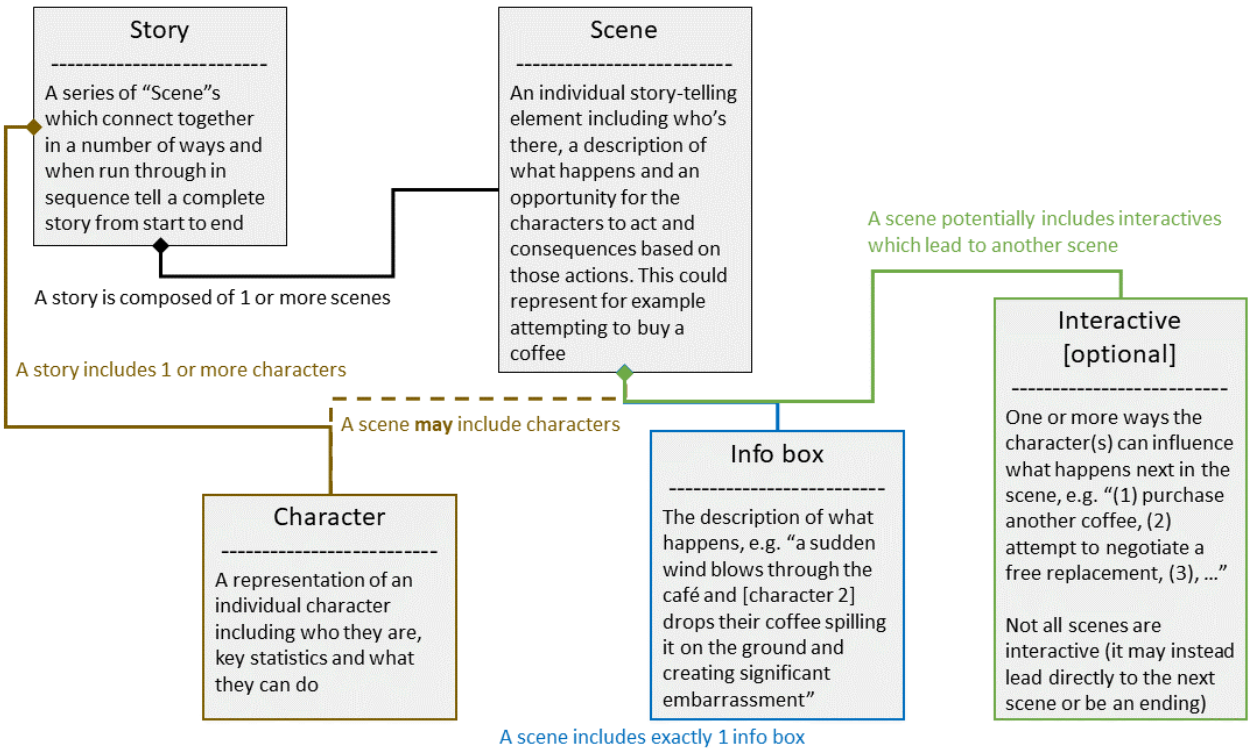
This assignment will have you and your team constructing a system to allow the user to run through a 'choose-your-own-adventure' style game with the individual elements of this drawn from text file(s)

Your task

By the completion of this assignment you should have a class structure representing

- a "story"
    - composed of "scenes" including
        - who ("character"(s)) is/are present in the scene,
        - a description of what happens,
        - some ways for the characters to interact in the scene,
        - links to the next scene (based on what the user chooses where interactives exist).
- a "character"
    - who have some key personal attributes and skills relevant to the story or scene.

The exact relationship is also shown in the diagram below



together with a program that populates these based on a given file and runs through the story for the

user.

Key steps

1. create the story and character classes as defined,
2. read in character details from a txt file,
3. read in a story into your class structure from a txt file,
4. allow the user to move from one scene to another,
5. determine level of success with an interactive and change which scene is visited next using character 1's abilities,
6. revise interactive components to allow groups of characters to participate,
7. [advanced] revise the presentation and interactive components to use specific characters using character codes.

Supporting Information 1: Stories

Story format

The stories used in this program are intended to be a form of interactive story-telling and can be thought of as a form of choose-your-own-adventure with a block of narrative/exposition followed by an opportunity to interact or change the story (or in some cases it may lead to another expositional block if it's felt the block is too long). Unlike the base choose your own adventure type stories, it is intended that a group might play through this together each attempting to take the role of a particular character in the story and agreeing on the choice to progress the story forwards.

For those who have played interactive story-telling games, table top role play games or what have you such as Fate, Dungeons and Dragons, Vampire the Masquerade, Everyone is John, etc. you may already be familiar with this particular form of story-telling (though the system we are using here doesn't fully or accurately reflect any particular system known to the assignment author).

These stories do not have to obey the laws of physics, or anything like that, it's more about having an interesting story and the suspension of disbelief is expected.

Below you will find the file format for a story together with an example of (part of) such a story that could apply to the system you will create here

Story File format

```
----
ID
text goes here
possibly multiple lines
====
1. [skillOrAttribute Difficulty] text --ID -ID +ID ++ID
----

----
ID2
text for 2
====
1. [skillOrAttribute Difficulty] text -ID +ID ++ID
2. text --ID -ID +ID ++ID
...
N. [skillOrAttribute Difficulty] text -ID +ID
----

----
... etc.
----
```

Above is an example of the type of format to expect.

- An individual scene begins and ends with a sequence of four dashes (`----`).
- Two scenes might be separated by different amounts of newlines (in between two `----`).
- After the first set of dashes the ID of the scene is given
    - the very first scene in a story uses the ID of `S` (which represents **S**tart)
    - the ending scenes (remembering there can be multiple endings dependant on user choices) follow the format:
        - E[type: * for 'good' and ~ for 'bad'][positive number] e.g. `E*1` or `E~5` which represent the first 'good' ending and the fifth 'bad' ending respectively
    - Otherwise, all other scenes simply use a positive number as their ID
    - other than the ID nothing else should appear in this line
    - no IDs should be repeated, these need to be unique to uniquely identify scenes and allow transitions between them
- after the ID line, all remaining text until a sequence of four equals signs (`====`) is treated as the description of the scene and what's happening
    - characters can be included in this text by using the format `{CN}` where N would be the position of that character's name in the list of characters
    - e.g. `the coffee cup was too hot for {C2} to hold...` if the characters were Albus and Bathilda then in the text displayed to the user they would see `"the coffee cup was too hot for Bathilda to hold..."`

> ⚠ extracting and replacing character names in scenes is fairly involved and you may wish to leave this out until everything else is working

- after the four equals signs (`====`), we have individual options for the user to select
    - the format for these is `Num. [skill/attribute difficulty] textDescription --ID_of_very_bad_outcome -ID_of_bad_outcome +ID_of_good_outcome ++ID_of_very_good_outcome`
        - e.g. `2. [body 3] Make a run for it -E~1 +E*2`
        - in this case, if a **body** check meets the score of 3 it goes to scene `E*2` or otherwise `E~1`
    - the `Num.` is included to aid readability of the file, it should always be provided however when displaying an option to the user you can always infer it from the order of the lines of options if this isn't read directly
    - the skill/attribute is optional and implies that particular skill or attribute is used to determine the outcome
        - where this is provided it must match the name of an existing skill/attribute and should include a difficulty value afterwards which must be an integer
    - the text description is **required** and explains what the player is (trying) to do to move to the next scene
        - as with other text the `{CN}` format can be used to reference character names
        - You can assume that the text description does not contain the characters `'.', '[',`

`']', '-'` or `'+'`.

again replacing character names is fairly involved and you may wish to leave this out until everything else is working

    - finally the IDs for the following scene are **required** (between 1 and 4 must be provided)
        - `--` represents a 'very bad' outcome -- if the skill or attribute check is an overwhelming failure this is the scene to transition to
        - `-` represents a 'bad' outcome -- if the check is a regular failure (not overwhelming) then we go to this ID's scene
        - `+` represents a 'good' outcome -- the check succeeded but not overwhelmingly
        - `++` represents a 'very good' outcome -- the check was a overwhelming success
        - the ordering of outcomes is not fixed (e.g. a `+` could appear before a `-`) instead the `+` or `-` character tells us about the nature of the outcome
        - what occurs when not all of these outcomes are represented is explained in a later section (but essentially that option is treated as the closest available option)
    - if not options are provided for a scene the story should be at an end point (e.g have an ID beginning with E)
- after every individual option is represented four dashes (`----`) signal the end of the scene definition
- the next scene definition begins with its own four dashes (`----`) but an empty line is included between for ease of readability

Sample story

Characters: Tinashe, Mehr, and Ujarak

Tinashe, ***Mehr***, and ***Ujarak*** are sitting in the little chicken café together after happily having submitted their assignment 3. This is the most convenient spot for them and was where they worked on the assignment together. Feeling their caffeine levels dropping below optimal, ***Ujarak*** heads to the counter and offers to buy everyone a coffee. Many seconds pass while waiting in line (at least seven!) before they reach the front only to discover they left their wallet at home...

==========

1. ***Ujarak*** decides to use their **diplomacy** skills to request ask for a freebie
2. ***Ujarak*** decides to draw on all their internal **acumen** to *will* a coffee into existence
3. ***Ujarak*** decides to use their **acrobatics** skills to dash home and return with their wallet before the other patrons are the wiser
4. ***Ujarak*** decides to give up and return to the table

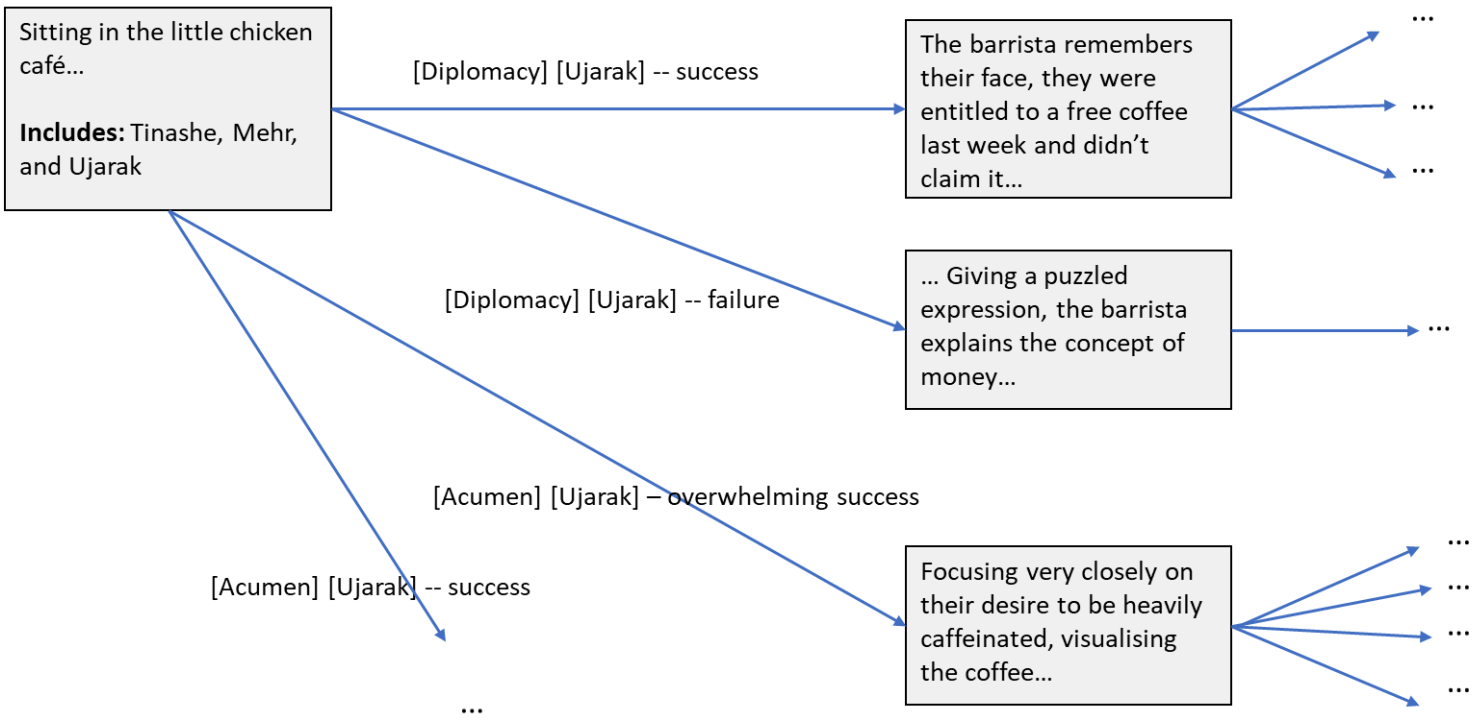
[we assume Ujarak decides on option 2 here and are remarkably successful]

Focusing very closely on their desire to be heavily caffeinated, visualising the coffee before them and drawing on all the mental faculties and strength of will they can muster ***Ujarak*** speaks their desire to the universe "Oh great provisioners of stimulant beans and their associated beverages, I call upon thee. Share with me your bounty!" They feel a growing solid form of cup within there hand. Minutes pass as they continue to start at their cupped hands and reiterating their desire. The distinct aroma of fresh-brewed coffee wafts its way into ***Ujarak***'s nostrils, they feel the warmth between their hands and low and behold in front of their eyes, their will is actioned. ***Ujarak*** returns to the table with their new coffee and a grin on their face. ***Mehr*** gives ***Ujarak*** a questioning look, "Something wrong with their machine?". Consequently...

==========

1. ***Mehr*** draws upon their **language** skills to determine whether ***Ujarak***'s empty hands represents an element of a sign-language they are familiar with
2. ***Tinashe*** use their **acrobatics** skills to get to the counter and order for themselves before they close
3. ...

This story is also represented in the image below



each scene leads to another scene through the outcomes of the interactive component

The file representation of this story could be as below (click to expand)

```
____
S
```

{C1}, {C2}, and {C3} are sitting in the little chicken cafe together after happily having submitted
====
1. [diplomacy 5] {C3} decides to use their diplomacy skills to request ask for a freebie -1 +2
2. [acumen 4] {C3} decides to draw on all their internal acumen to *will* a coffee into existence +
3. [acrobatics 3] {C3} decides to use their acrobatics skills to dash home and return with their wa
4. {C3} decides to give up and return to the table -E~1

1
Giving a puzzled expression, the barrista explains the concept of money...
====
... etc

2
The barrista remembers their face, they were entitled to a free coffee last week and didn't claim i
====
... etc.

3
Focusing very closely on their desire to be heavily caffeinated, visualising the coffee before them
====
1. {C2} draws upon their language skills to determine whether {C3}'s empty hands represents an elem
2. [acrobatics 3] {C1} use their acrobatics skills to get to the counter and order for themselves b
... etc

E~1
Such a shame, it appears our collectives desire to be caffeinated shall remain forever... unresolve
====

Supporting Information 2: Characters

Character format

As part of the program, it is intended that the 'choose-your-own-adventure' style story-telling involves multiple protagonists who can influence how the story progresses.

For the purposes of this assessment you can assume one human user will be deciding which option to take to go from one scene to the next however you could imagine a scenario (and perhaps this might be more fun) where there are a few folks sitting together with one person 'driving' but each human representing one of the playable characters (protagonists) and they would decide together which option to choose.

Character attributes

Each playable character in the story has different aspects to them which can influence where they are more likely to succeed or fail in an interactive component, there are several different aspects which characters may have:

Attributes

- Acumen -- this represents general logical reasoning, memory and intuition,
- Body -- this represents what one can do generally with their physical form such as fine motor skills, reaching something, physical strength,
- Charm -- this represents their ability to understand others and influence their decision making such as making a diplomatic request to someone, inferring how they are feeling, etc.

Each character will have a certain number of points in each of these three:

- the minimum in any is 1, the maximum 4,
- overall, the sum of these scores should add to 7,
- e.g. 4,2,1 and 1,3,3 are valid distributions while 7,0,0 and 4,4,2 are not.

Skills

- diplomacy
  - convincing others through words and temperament
  - relies on **charm**
- investigation
  - critically examine a situation to understand what's going on
  - relies on **acumen**
- medicine
  - knowledge and practice of health care

    - relies on **acumen**
- language
    - ability to communicate meaning in different forms
    - relies on **charm**
- acrobatics
    - the ability to move quickly and skillfully
    - relies on **body**
- craft
    - use of fine motor skills to build or interact with the physical world
    - relies on **body**

Each of these skills rely on a single attribute to mainly contribute to success with them. In addition, every character is **proficient** in exactly one of these, making them more likely to succeed than another (see the next section for more details). Both of these result in a score for a skill being the relevant attribute +2 if they are proficient in that skill.

File format

Below is the file format holding a set of characters with some examples

```
[character name]
A[num] B[num] C[num]
Di In Me* La Ac Cr
----
[character name]
A[num] B[num] C[num]
Di In Me La Ac Cr*
----
etc.
```

where the asterisk (*) represents the skill the character is proficient in. You can assume that the order of the lines and the order of the attributes and skills is always the same.

Sample characters

Hubert Mann

▶ Expand

Rob Boht

▶ Expand

Anne Uther

▶ Expand

Corresponding file

```
Hubert Mann
A4 B1 C2
Di In* Me La Ac Cr
----
Rob Boht
A1 B3 C3
Di In Me* La Ac Cr
----
Anne Uther
A2 B2 C3
Di In Me La* Ac Cr
```

Supporting information 3: Influencing an outcome

Most scenes have an interactive stage where one or more characters can influence which scene is reached next.

Once the option is chosen by the user, success is determined by the difficulty of the option, the skill / attribute level of the (relevant) character (including proficiency) and an element of random chance.

The random element is determined by randomly selecting one of (-1, 0, 1) three times (in a uniform manner, which means each of the three option is equally likely) and summing the result.

Difficulty levels

Different final scores represent a particular level of difficulty

- Fantastic = 6
- Great = 4
- Fair = 2
- Average = 0
- Terrible = -2

and the author of the story would decide what score would have to be reached to count as a success (e.g. perhaps one author might treat the **investigation** option of "find free food on campus" as something requiring a 'great' effort while another author might consider this simply 'average').

Provided the score for an outcome in the file is a whole number this is considered valid.

Levels of success

If the final score for an attribute / skill attempt

- **equals** the difficulty level this is considered a **success** (as is **exceeding** the difficulty by **up to 2 points**)
- **exceeds** the difficulty level by **3 or more points** this is an **overwhelming success**
- **falls under** the difficulty level by **1 to 3 points** this is a **failure**
- **falls under** the difficulty level by **4 or more points** this is an **overwhelming failure**

Sample calculations

For the option below

```
[acumen 4] draw on all their internal acumen to *will* a coffee into existence ++3 +4 -1
```

and Anne Uther with Acumen: 2 (as defined below)

```
Anne Uther
A2 B2 C3
Di In Me La* Ac Cr
```

overwhelming failure

▶ Expand

failure

▶ Expand

success

▶ Expand

overwhelming success

▶ Expand

1. Story and Character classes

Your task

- Prepare a file called **`story_class_structures.py`** which **contains** (at least) the two classes defined below.
- Create an `if __name__=="__main__":` block in the file which instantiates the story and character classes to represent a sample story and character (a hard-coded list is fine for and will allow you to test your work)

Ensure that your class and function names match the ones we provide, as the automated marking will use these names for testing. You can add additional properties and methods to the classes as well as create entirely new supportive classes as needed to simplify or support your code

Classes and mapping functions to them

`Character` class

▶ Expand

`Story` class

▶ Expand

Story class string format

▶ Expand

2. making a check and advancing the story

Your task

- Revise your `story_class_structures.py` file to add three additional methods.
    - one to the Character class and the others to the Story class
- Create an `if __name__=="__main__":` block in the file which instantiates the story and character classes to represent a sample story and character (a hard-coded list is fine for and will allow you to test your work)

Ensure that your class and function names match the ones we provide, as the automated marking will use these names for testing. You can add additional properties and methods to the classes as well as create entirely new supportive classes as needed to simplify or support your code

You will need

- the randint function of the random python package

`Character.make_check(self,skill_or_attribute_name,difficulty,override_random)`

▶ Expand

`Story.show_current_scene(self)`

▶ Expand

`Story.select_option(self,option_number,override)`

▶ Expand

3. Working with files

Prepare a python file called **`choose_your_own_adventure.py`** which

1. defines the **`read_file`** function as defined below

2. runs the **`read_file`** function in your **`if __name__ == "__main__"`** block using it to create an instance of the Story class from a given story and character file

3. interacts with the user (within that **`if __name__ == "__main__"`** block) to allow them to progress through the story selecting options as they go

- in essence alternating between **`show_current_scene()`** and **`select_option()`** with the option number depending on the user's choice (and ensuring the random override is `None`)

Testing your program

You may find it helpful to use (as a starting point) these sample files below holding the scenes and characters file formats respectively

You will need

- your code to prior tasks

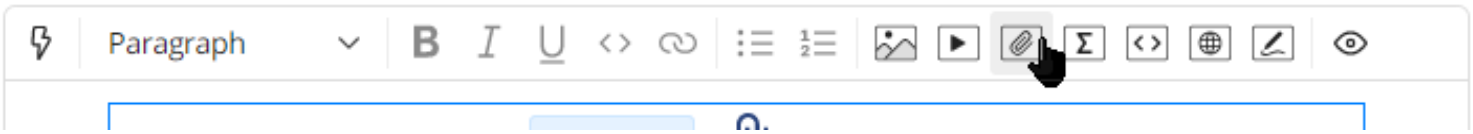
`read_file(filename)`

▶ Expand

[Just for fun] sharing stories with your peers

This is **not assessed** but is a nice opportunity to have a bit of fun and share with your peers -- this will also double as extra test-cases for you and your peers.

Write up some stories of your own (following the format given in supporting information 1) and post them in the Ed discussion under the channel **A4 stories** using the paper-clip attach file option



in the title of your post you can put the name of the story, the parts required to be complete for the story to work and the number of characters intended

e.g. "Oak the tree person's grassroots campaign", part 1-3 required, 1 character

You are encouraged to be creative and design stories that interest you and/or draw inspiration from stories from your family and community as you see fit.

By the way, as long as we're talking about story-telling, it would be remiss not to mention the many Indigenous Australian peoples (who's lands many of us are standing upon), with their rich oral history and tradition (within which story-telling is central) and being one of the longest continuing cultures in the world. If you're interested, here is a video with artist Maree Clarke (prepared for the NGV as a learning resource) talking about the possum skin cloaks which is itself a medium of story-telling and which is super interesting to myself as a computer science educator as I see within this a tradition of teaching both an art and algorithm together and this is how I would wish you to see programming and computer science yourself.

4. Individual and group checks

Your task

Create a new file `story_class_structures_bestchar.py` which:

- imports the Story and Character class from your base file
- includes a `StoryBest` class which inherits from the `Story` class and
    - adds the `select_character_for_check` defined below
    - revises the `select_option` method as defined below

You should also revise your `choose_your_own_adventure.py` file to use `StoryBest` instead of `Story` which will allow you to work with multi-character stories more meaningfully

Ensure that your class and function names match the ones we provide, as the automated marking will use these names for testing. You can add additional properties and methods to the classes as well as create entirely new supportive classes as needed to simplify or support your code

You will need

- your code from previous parts

`StoryBest.select_character_for_check(self,skill_or_attribute_name)`

▶ Expand

`StoryBest.select_option(self,option_number,override)`

▶ Expand

[Not assessed] Your thoughts on the program

Think about your efforts so far over the assignment and the different topics and approaches you've encountered...

Question 1 *Submitted May 21st 2022 at 1:36:30 pm*

How well do you think you have understood the concepts and ideas that were relevant to this assignment?

- ○ Not well
- ◉ okay
- ○ well

Question 2

What's something new you learned?

No response

Question 3

What did you find most challenging?

No response

Question 4

What would you like to do differently moving forwards?

No response

END OF FIT1045 PART

5. Advanced component

Your task

Create a new file `story_class_structures_chosenchar.py` which:

1. imports the Story and Character class from your base file
2. includes a `StoryChosen` class which inherits from the `StoryBest` class and
    - revises the `select_character_for_check` as below
3. revises how the `StoryChosen` class is instantiated to replace character codes (e.g. {C2} with "My Name") in the scenes as per the details in Supporting Information 1

Note: step 3 may involve going back to revise your original Story class definition. Be sure to save a copy of the original working version if this is the case

You will need

- your code from previous parts

`StoryChosen.select_character_for_check(self,skill_or_attribute_name, scene_chars, option_char)`

▶ Expand

END OF FIT1053 PART