12 Lists

Quite often we want to store a whole set of items (for example, names). Having a variable for each one is not very helpful (particularly if we want to add more names to the lists or remove some names). Suppose we have a small class of 6 students and we want to store all of their names.
There names are: Agatha , Melinda, Harriet, Hine, Dandelion and Te Aroha
To create a list we use square brackets and commas.

```
# each item in the list has an index number 0,1,2,3,4,5
# the first item is at index 0 (Agatha)
# the length of the list is 6 (because it has 6 items in it)
# but the last index number is 6-1 = 5 ( because the counting starts from 0 )

my_class = ["Agatha" ," Melinda", "Harriet", "Hine", "Dandelion" , "Te Aroha"]
```

Listing 18: Python example

To manage lists, there are a large range of methods that can be used.
See the examples below.

```
my_class = ["Agatha" ,"Melinda", "Harriet", "Hine", "Dandelion" , "Te Aroha"]

#print out a name at a specific index (use the square bracket notation)
# this prints out the fourth person in the last
print(my_class[3])
print()
# print the list out "informally"
print(my_class)
print()
# print using a loop (this is the usual way to do it)
# this a shorthand way of doing it,
# where the "x" is, we can use anything we like and it
# will hold the value on each iteration of the loop
for x in my_class:
    print(x)
print()
# find out how long the list is (you often need to do this)
class_length = len(my_class)
output = "There are {} students in the class\n".format(class_length)
print(output)
# sort the list
my_class.sort()
print(my_class)
print()
# shuffle the list
# for this we need the random module(should be up the top)
import random
random.shuffle(my_class)
```

```
print(my_class)
print()
# add a new name to a list
my_class.append("Phoebe")
print(my_class)
print()
# check if someone is in the list
check_name = "Jane"
if check_name in my_class:
    print("{} is in the class".format(check_name))
else:
    print("{} is not in the class".format(check_name))
print()
# find the index number of a particular name (where is Harriet??)
index_num = my_class.index("Harriet")
print("{} is at index number {}".format("Harriet" , index_num))
print()
# remove someone from the list
my_class.remove("Harriet")
print(my_class)
print("{} is no longer in the list".format("Harriet"))
```

Listing 19: Python example

See also:

- Python Lists: https://www.w3schools.com/python/python_lists.asp
- Python List Methods: https://www.w3schools.com/python/python_lists_methods.asp

13 Lists and Loops

13.1 Creating and modifying a class group

Scenario:

Create a simple program that asks the user to enter the names for students in a class.

Once the names have been entered it prints out the names as a numbered list.

Extension:

One the names have been entered and listed, ask the user if they would like to modify or remove any of the names. (and if they do allow them to do it)

13.2 Lucky Unicorn Game

13.2.1 Program

Scenario:
You have decided to create a fun game to raise money for the charity Doctors without Borders.
You will set up your computer at lunchtime and players will pay to play.

Here are the rules:
Users pay an initial amount at the start of the game.

- The cost should be $1 per round and users should press <enter> to play.
- The computer should then generate a token that is either a zebra, horse, donkey or unicorn.
- This should be displayed to the user (as text).
- If the token is a unicorn, the user wins $5, if it is a zebra or horse, they win 50c and if it is a donkey then they don't win anything.

The maximum amount of money that students can spend on the game is $10 per session.

- The game should allow players continue / quit provided they have not lost all of their money.
- It should supply appropriate feedback so that the user knows how much money they have won / lost each round and how much money they have left.

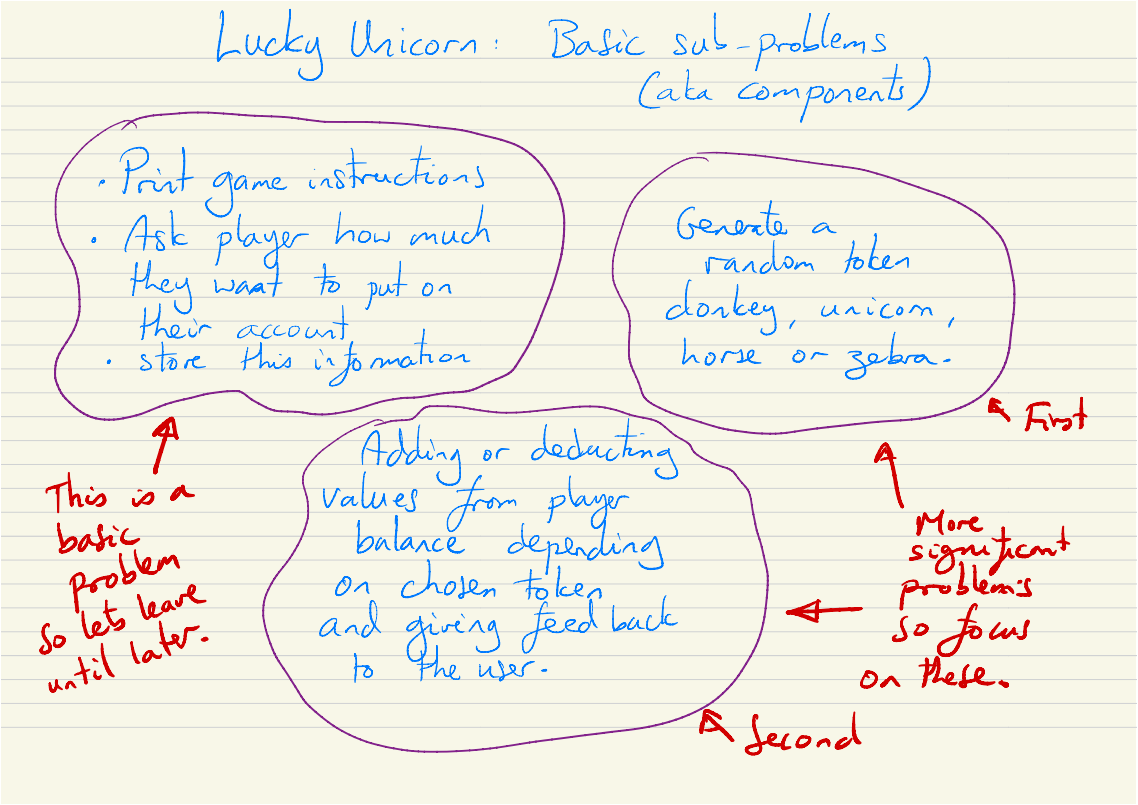
Once students have no more money, the game should end (although players do have the option of quitting while they are ahead).

Variations (optional) Once you have a working game, you are welcome to develop the outcome further.
Here are some variations you could consider...

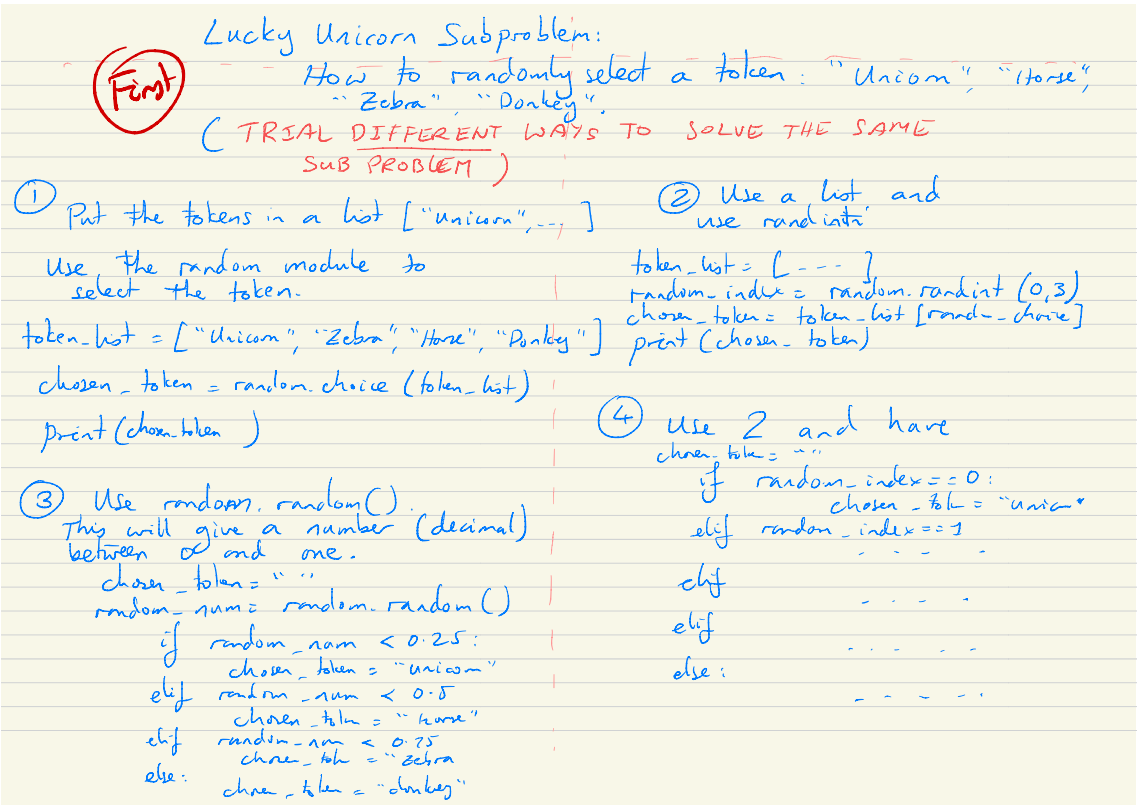
- Change the tokens / context (so rather than having zebra, horses, donkeys and unicorns the game could involve other items)
- Allow users to bet more than $1 per round and adjust the pay-outs proportionally
  (be careful to set up you game so that the house has a long term advantage)

13.2.2 Planning: Iterative Processes

- Break the problem down into parts (Draw some kind of diagram)
- Each of the parts can be called a "component" of the program or a "sub-problem".
- You need plan and test code for each of these "sub-problems".
  - Ideally (and this must happen somewhere), you need to look at "trialling" different ways to solve the sub-problem), with the idea being that you can trial the different ways and choose the best (and also look to ways of improving).
    This needs to be captured in the planning documentation for the iterative processes assessment.
- Combine your code into a fully working program (your first version)
  - Test the program and document the test (gridded test plan)
  - Write a short reflection on what works and what could be improved upon.
- Plan the next version
  - Test the program and document the test (gridded test plan)
  - Write a short reflection on what works and what could be improved upon.
- Repeat this planning and testing process.
- If new sub-problems appear , then work on them in the same way as referred to above



Breaking the problem down into components



Planning options for the first sub-problem

Component (aka sub-problem) trials for randomly selecting a token.
With ouputs to test that they are working.

get_token_1.py - /Users/khouripa/Documents/DTech/year11_2020/co...

```
# Trial idea 1
import random
token_list = ["unicorn", "horse", "zebra", "donkey"]
chosen_token = random.choice(token_list)
print(chosen_token)

#loop test
# to make sure all the tokens are getting chosen ,
# roughly equally
count = 0
while count < 100:
    chosen_token = random.choice(token_list)
    print(chosen_token)
    count += 1
```

Ln: 9 Col: 2

```
zebra
unicorn
donkey
donkey
donkey
unicorn
zebra
donkey
donkey
donkey
unicorn
```

get_token_2.py - /Users/khouripa/Documents/DTech/year11_2020/code_activities/l...

```
#Trial idea 2
import random
token_list = ["unicorn", "horse", "zebra", "donkey"]
random_index = random.randint(0,3)
chosen_token = token_list[random_index]
print(chosen_token)

#loop test
# to make sure all the tokens are getting chosen ,
# roughly equally
count = 0
while count < 100:
    random_index = random.randint(0,3)
    chosen_token = token_list[random_index]
    print(chosen_token)
    count += 1
```

Ln: 7 Col: 0

```
donkey
zebra
unicorn
donkey
donkey
horse
donkey
donkey
horse
horse
horse
horse
```

get_token_3.py - /Users/khouripa/Documents/DTech/year11_2020/code_activities/luckyunicorn/sub_pr...

```
#Trial idea 3
import random
chosen_token = ""
random_num = random.random()
if random_num < 0.25:
    chosen_token = "Unicorn"
elif random_num < 0.5:
    chosen_token = "Horse"
elif random_num < 0.75:
    chosen_token = "Zebra"
else:
    chosen_token = "Donkey"

print(chosen_token)
```

Ln: 17 Col: 0

```
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_act
ests/get_token_3.py
Unicorn
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_act
ests/get_token_3.py
Horse
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_act
ests/get_token_3.py
Horse
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_act
ests/get_token_3.py
Donkey
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_act
ests/get_token_3.py
Donkey
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_act
ests/get_token_3.py
Donkey
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_act
ests/get_token_3.py
Zebra
>>>
```

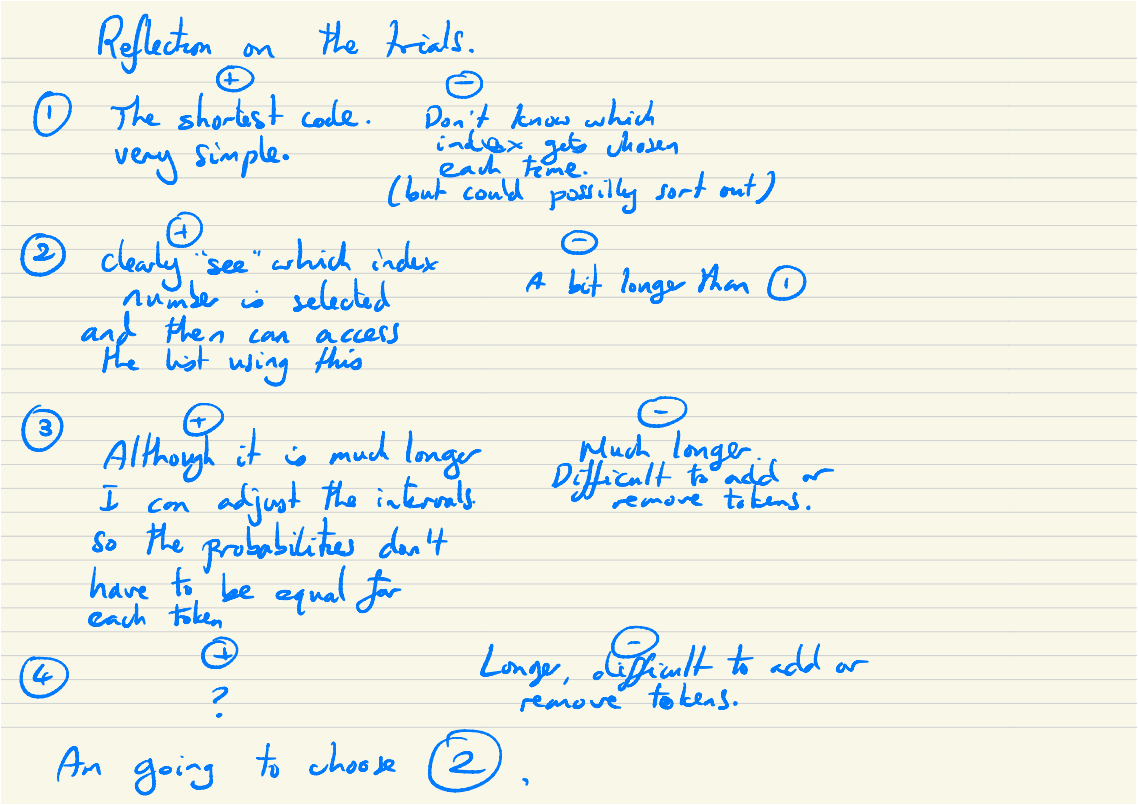
get_token_4.py - /Users/khouripa/Documents/DTech/year11_2020/code_activities/luckyunicorn/sub_pr...

```
#Trial idea 4
import random
chosen_token = ""
random_num = random.randint(0,3)
if random_num == 0:
    chosen_token = "Unicorn"
elif random_num == 1:
    chosen_token = "Horse"
elif random_num == 2:
    chosen_token = "Zebra"
else:
    chosen_token = "Donkey"

print(chosen_token)
```

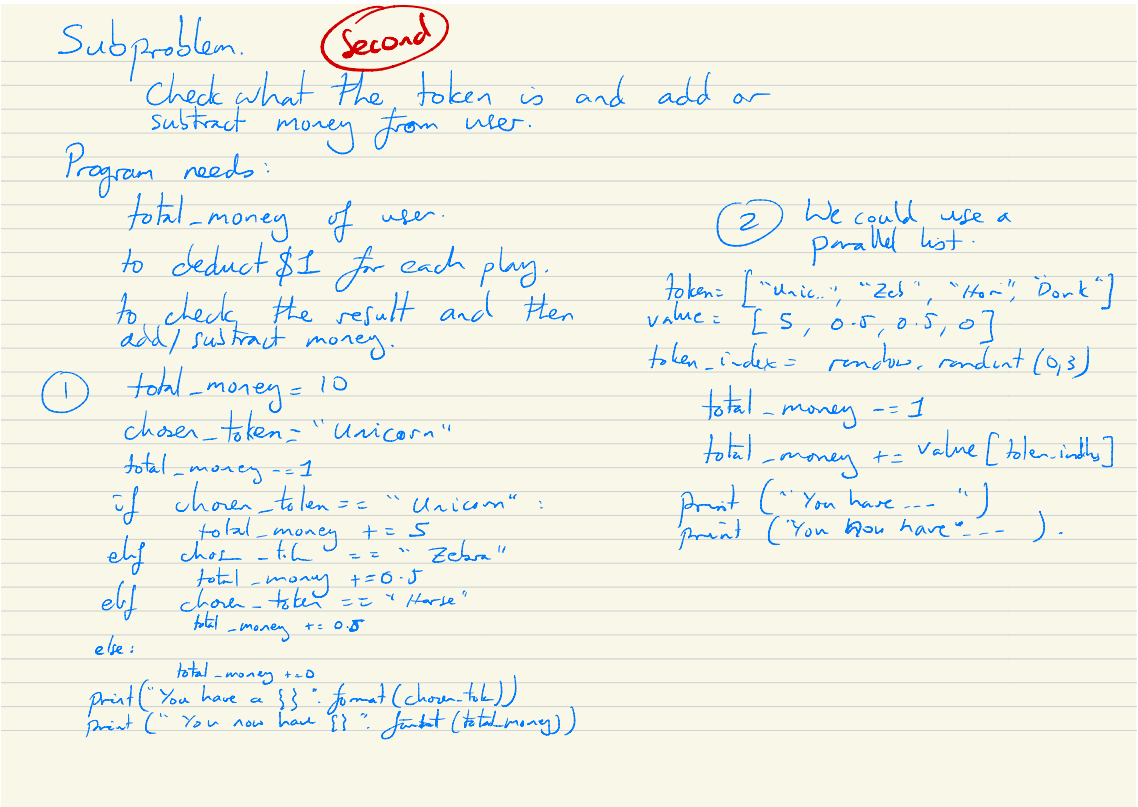
Ln: 15 Col: 4

```
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activ
ests/get_token_4.py
Donkey
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activ
ests/get_token_4.py
Zebra
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activ
ests/get_token_4.py
Zebra
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activ
ests/get_token_4.py
Donkey
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activ
ests/get_token_4.py
Unicorn
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activ
ests/get_token_4.py
Horse
>>>
```

Trialling different ways to solve the same sub-problem



A reflection on the trials and a choice



Planning options for the second sub-problem

Component (aka sub-problem) trials for adding or deducting player money and feedback to player.

add_subtract_feed_back.py - /Users/khouripa/Documents/DTech/year11_2020/code_activities/luckyunicorn/sub_...

```
total_money = 10
chosen_token = "Zebra"

total_money -= 1
if chosen_token == "Unicorn":
    total_money += 5
elif chosen_token == "Horse":
    total_money +=0.5
elif chosen_token == "Zebra":
    total_money +=0.5
elif chosen_token == "Donkey":
    total_money+=0

print("You got a {}".format(chosen_token))
print("You now have ${:.2f}".format(total_money))
```

Ln: 2 Col: 21

```
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activities/luckyuni
ests/add_subtract_feed_back.py
You got a Unicorn
You now have $14.00
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activities/luckyuni
ests/add_subtract_feed_back.py
You got a Donkey
You now have $9.00
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activities/luckyuni
ests/add_subtract_feed_back.py
You got a Horse
You now have $9.50
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activities/luckyuni
ests/add_subtract_feed_back.py
You got a Zebra
You now have $9.50
>>>
```

add_subtract_feed_back_1.py - /Users/khouripa/Documents/DTech/year11_2020/code_activities/luckyunicorn/su...

```
import random
total_money = 10
token_list = ["unicorn", "horse", "zebra", "donkey"]
values = [5, 0.5, 0.5, 0]
token_index = 3

total_money -=1
total_money +=values[token_index]


print("You got a {}".format(token_list[token_index]))
print("You now have ${:.2f}".format(total_money))
```

Ln: 5 Col: 15

```
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activities/luckyunicorn/sub_proj
ests/add_subtract_feed_back_1.py
You got a horse
You now have $9.50
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activities/luckyunicorn/sub_proj
ests/add_subtract_feed_back_1.py
You got a unicorn
You now have $14.00
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activities/luckyunicorn/sub_proj
ests/add_subtract_feed_back_1.py
You got a unicorn
You now have $14.00
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activities/luckyunicorn/sub_proj
ests/add_subtract_feed_back_1.py
You got a zebra
You now have $9.50
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activities/luckyunicorn/sub_proj
ests/add_subtract_feed_back_1.py
You got a zebra
You now have $9.50
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activities/luckyunicorn/sub_proj
ests/add_subtract_feed_back_1.py
You got a donkey
You now have $9.00
>>>
```

Improvement after the first trial

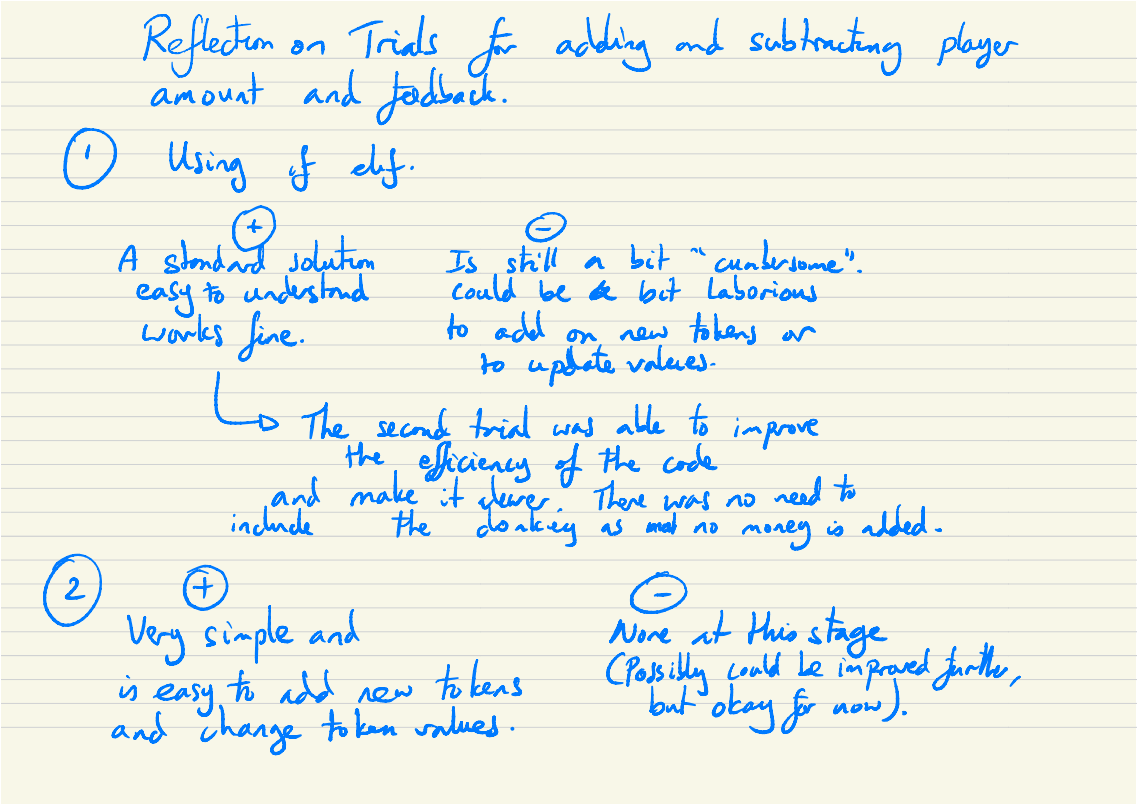
add_subtract_feed_back-a.py - /Users/khouripa/Documents/DTech/year11_2020/code_activities/luckyunicorn/sub_problem...

```
# after the first trial of this idea
# I modified it to simplify the
# conditions so it is now mush more efficient

total_money = 10
chosen_token = "Unicorn"

total_money -= 1
if chosen_token == "Unicorn":
    total_money += 5
elif chosen_token == "Horse" or chosen_token == "Zebra":
    total_money +=0.5


print("You got a {}".format(chosen_token))
print("You now have ${:.2f}".format(total_money))
```

Ln: 6 Col: 23

```
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activities/luck
ests/add_subtract_feed_back-a.py
You got a Zebra
You now have $9.50
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activities/luck
ests/add_subtract_feed_back-a.py
You got a Horse
You now have $9.50
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activities/luck
ests/add_subtract_feed_back-a.py
You got a Donkey
You now have $9.00
>>>
= RESTART: /Users/khouripa/Documents/DTech/year11_2020/code_activities/luck
ests/add_subtract_feed_back-a.py
You got a Unicorn
You now have $14.00
>>>
```

Trialling different ways to solve the same sub-problem (the second)



A reflection on the trials and a choice