Code testing

Contents

[1. User input 2](#_Toc8835988)

[User input name: 2](#_Toc8835989)

[User input surname: 2](#_Toc8835990)

[User input class: 2](#_Toc8835991)

[User input answer: 2](#_Toc8835992)

[2. Testing strategy: 3](#_Toc8835993)

[3. Testing name input: 3](#_Toc8835994)

[4. Testing answer input 4](#_Toc8835995)

[5. Pseudo code 5](#_Toc8835996)

[5.1 Old pseudo code: 5](#_Toc8835997)

[5.1.1 Explanation 5](#_Toc8835998)

[5.2 New pseudo code: 6](#_Toc8835999)

[5.2.1 Explanation 6](#_Toc8836000)

[6. The game 7](#_Toc8836001)

[6.1 How it works 7](#_Toc8836002)

[6.2 Limitations 7](#_Toc8836003)

[6.3 Implementations 7](#_Toc8836004)

[6.4 Difficulties 8](#_Toc8836005)

# User input

## User input name:

When the user inputs a name, it will look at table to see if any of the characters are included in this table. If this is not the case it will allow the user to input this name. If not, it will tell the user the name contains possible illegal characters or the name is too long and the user has to re input the name.

The table it looks at includes these characters: '[@!#$%^&\*()<>?/\\\|}{~:0-9`=+;]'

## User input surname:

Surname follows the same table as name. The only difference is that surname allows longer names than name. 20 instead of 15.

## User input class:

Class has a different table because it might need characters which name and surname don’t allow. For example numbers. It works the same as name and surname just a different table

Table in question: '[@!#$%^&\*()<>?/\\\|}{~:=+`;]'

## User input answer:

The user input works different from name, surname and class. Instead of looking at a table, it checks if it can convert it to an integer. If this is possible it will allow the user to input the value. If not, it will re ask the question to the user and tell that the answer before was not allowed.

# Testing strategy:

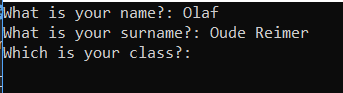
The strategy for testing will be to just input as many inputs as I can. So I’ll try Olaf() and Olaf= if any of these are allowed which they shouldn’t I’ll fix it by adding that character to the table. If have also asked friends to play the game and tell me if they managed to break any of it. This is so if I forgot to check something. Someone else might try it. There isn’t much else the user can do to try and break it Because it can only input data. (Unless the user goes into the code and chances something)

# Testing name input:

**Names, surname and class input:**

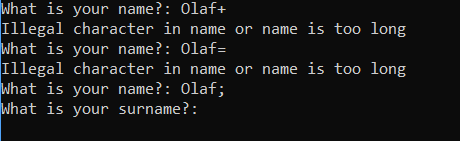
**Works:**

* Olaf= (Not intended fixed now)
* Olaf; (Not intended fixed now
* Olaf+ (Not intended fixed now)
* Olaf-
* Olaf
* Olaf or
* Olaf’s
* Oléf
* Neß
* Olaf1 (Works for class only)



**Doesn’t work:**

* Olaf()
* Olaf1
* Olaf/
* Thisisareallylongnamethatkeepsongoing (Too long)
* Olaf&
* Olaf^
* Olaf$
* Olaf%
* Olaf#
* Olaf@
* Olaf!
* Olaf\
* Int(1)
* Str(a)
* “Blank space”
* “Enter”



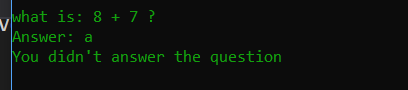
# Testing answer input

**Works:**

* 1
* -2

**Doesn’t work:**

* A
* a
* int(1)
* ()
* @
* 1.0 (Decimal point is not needed



# Pseudo code

## Old pseudo code:

1. Put out a window for user input and output
2. put box for user input in window
3. Gain user name and class
4. Check date
5. Save user data + date
6. Display story text
7. Display progress bar start timer
8. Run random gen for questions
9. display questions and progress bar.
10. Check if questions is correct or not
11. save if correct or not per questions
12. Loop to line 9 till x questions have been answered.
13. Tell user correct and incorrect and if they made the end or not.
14. (Check for possible illegal characters)
15. upload all user data to database
16. close program.

### Explanation

The old pseudo code is from when I wanted to use pygame to display all the questions and make the user input text into there. After playing around with pygame I found out that it isn’t easy to make the user input text and seeing the deadline come closer I decided to use the console python gives you to make it that way. Changing it into a text based game.

## New pseudo code:

1. Create database
2. Create tables
3. Create console for user to input data and receive data
4. Ask for users name
5. Check for illegal characters
6. Ask for users surname
7. Check for illegal characters
8. Ask for users class
9. Check for illegal characters
10. Input user data into database
11. Display fake loading bar
12. Wait for user input to continue
13. Clear console
14. Change console text colour
15. Display progress bar
16. Display explanation
17. Wait for user input to continue
18. Clear console
19. Display story
20. Wait for user input to continue
21. Clear console
22. start loop
23. { Display progress bar
24. start timer
25. Display question
26. Display input for user
27. Wait for user input
28. Check for illegal characters
29. Stop timer
30. Display correct or incorrect and time taken
31. Clear console }
32. Input data into database
33. Loop till x questions answered
34. Check for number correct answers
35. Display text for next level if correct answers is enough
36. Else display You lose with story
37. Loops 4 times
38. input all data into database
39. commit to database

### Explanation

I made this before I started on my code so I know what I roughly want the game to do and how it should work. In the end some more things ended up in the game like it displays art with the text.

# The game

## How it works

When the user starts the game it will first ask the user for a name, surname and class. Once the user has input this data it will give the user a quick explanation of what is expected from them. It then displays the story of the first level. The game has 4 levels in total each level has it’s own small story and story if you don’t make it. The game requires you to have ~70% correct for you to make it to the next level. Once this is all done the data will be transferred to a database where the teacher can see it.

## Limitations

* The game only works on Windows. The reason for this is because I use an os.sytem() function to clear the console. There are specific functions for the different OS’s. My knowledge of programming is not far enough to know how to check for the OS the user is running.
* You need a Python IDE to play the game. This is because it doesn’t have it’s own .exe and runs in the console of the Python IDE
* The name is limited to 15 characters. This is so the database doesn’t get filled with joke names that are really long surnames is limited to 20 characters. Same for class.

## Implementations

In my code I used functions and classes. The reason I did this, is because it’s easier to re-use the same line of code instead of having to write another line for the same code. I have used functions in python before so I have a rough understanding of how they worked. I did however had to learn how to give data to the function when you call it. Also how to return data from the function.

I also use separate text files to store my story and art that is used in the game. The reason for this is so it doesn’t fill up my code and create a big mess. It also makes it easy to change something about the story without having to open the code and find the line it is on.

The same goes for the code that makes and inserts data into the database. This is in a separate python file and sits in a class. This is also to not create a big mess of the code and is something that can be outside of the main code.

I used GitHub to store my code online so I can easily access my code from both my laptop and my main pc at home. This is also so I can easily track my progress of my code. Help people by showing parts of my code. Or getting help with some things. I did push all the code updates to the master because I kept versions of the old code on my pc locally.

## Difficulties

The hardest part of it all was the database part. Most because of how little information the IDE gives you when there is an error. This made it so you couldn’t just google an error and fix it. It also made it a bit harder because I was ill for a long time during the weeks we had lessons about the databases. And it also didn’t help that in the lessons we used mysql instead of sqlite3. The databases were mostly trail and error. Change a bit of code and run your program.

Another thing I found a bit difficult was the logic around functions and returning data from a function. The problem I had with this was that I though I had an understanding of the logic behind it but this was proven wrong. Once I played around a bit more with functions testing out if the way I thought it worked works. I found the way it’s suppose to work. Playing around with it a bit more showed me how the logic behind functions in python work. Because I thought that if you returned for example X. X would become a global variable with you can always call by just saying print(X)

This is not the case I learned that you need to assign the function to a variable for example. var\_X and then call the function. Everything that the function now returns is stored in var\_X. So the line of code for this would be. var\_X = functionName(data1,data2). Now if I want to print something that functionName returns I need to say. print(var\_X) or if it returns multiple data sets I need to say: print(var\_X[1])

Those were the things I found difficult. I have played around with python before so I already knew a lot about how it works. For example while loops or for x in range (0,1). I have already done how to call a function or dividing variables before.