Briefs.

Problem Statement

You are to select ONE of the problem statements to use in the development of your prototype.

1. Exam Timetable

You are aware that:

* Exams are important
* You are forgetful
* You can use a computer to help your memory Hence will develop a prototype for a program that will store data about your exams.

There will be several exams, and for each exam, you will need to specify:

* Date
* Morning or Afternoon
* Exam title
* Location (i.e. Room number)

You must:

* Be able to add an exam
* Be able to delete an exam
* Be able to edit an exam
* Use persistent storage (i.e. store the data in a file)
* Be able to show a list of exams

1. RPG Sheet Tracker ‘Dungeons and Dragons’ style role playing games are fun and engage the imagination. They also require users to keep track of a constantly changing series of statistics, items and points. It would be helpful if there was a simple interface that could record a character’s statistics, abilities and inventory. It would also be helpful if this could be updated easily as the game progresses.

There may be several characters. Each character will have the following statistics:

• Health

• Strength

• Agility

• Intelligence

• Up to three special abilities

• Up to five items in inventory

• One item in hand

• Gold

You must:

• Set each character a class and a name

• Be able to set statistics and items

• Be able to update statistics and items

• Be able to show all details, statistics and items for the characters

1. Fast-Food Self-Ordering Kiosk Gone are the heady days of cashiers asking if you want your order “supersized.” Not only has the infamous upgrade gone by the wayside, but cashiers at fast-food restaurants are becoming increasingly uncommon. McDonald’s started rolling out ordering kiosks at its US locations in 2015, and the chain hasn’t looked back since: by 2020, most of its 14,000 locations will have kiosks installed (Johnson, 2018). It plans to add the kiosks to 1,000 stores every quarter for the next two years, according to CNBC. However, a recent poll conducted by Business Insider’s partner MSN suggests that diners aren’t big fans of automated kiosks: 78% of customers said they would be less inclined to go to a restaurant that has automated ordering kiosks. The popular narrative is that kiosks and mobile ordering are here to take jobs and hours away from underpaid cashiers, ultimately saving companies money in the face of rising labour costs Data Create an ordering system for a take-away food shop. A GUI menu takes input from the operator (menu items are chosen from a series of drop-down lists, including quantity) and interacts with an array of food items. Each order has a space for entering the name associated with the order. The program stores orders in an array. The program allows for take-away and delivery orders and calculates the final cost of the order. The menu drives all aspects of the ordering system.
2. Pitch your own idea!

**Initial thoughts**

I like the idea of the exam one as I have some connection to it, and it is something I will actually be able to use, I do not play D&D or many RPGs so I have no use for it. When it comes to my own ideas I do not have anything currently. So for now, it is option 1.

Research.

**Final Idea**

Materials and components.

Languages

Scratch

Scratch is a very visual language. It is based off of JavaScript and can be used to efficiently create quick and easy programs. Images are easy to make and use, movement is easy and manipulation of code is easy as well. I also know the language well.

But where it leads in simplicity it lacks in functionality, its text input and output is relatively simple and it’s more complex lists and functions are lacklustre in the version we have on the school computers.

Python

Python is an easy text based program. Its structure makes it very easy to read as it is a very high level language I know the language very well and so does my teacher and classmates so any feedback I get from them they will have the limitations of the language in mind. It is extremely popular online and so there is a lot of resources to help.

As it is high level it is slow in more complex functions but as it is only a text based game this is not really a problem.

JavaScript

JavaScript is an extremely versatile language as it can be integrated into HTML and CSS which is what most of the internet uses to display information. As it can be integrated into CSS and HTML it is very customisable and has a lot of room to make anything. JavaScript’s libraries make the possibilities essentially endless (p5.js). Most of the internet uses it so there are a lot of resources available to help me.

While I know js well I do not know how to integrate it into HTML as confidently as I would like. While it is definitely something I could easily learn it adds another variable and stress into the mix where I want the only thing to limit my ideas is the stakeholders.

HTML and CSS

Most of the internet uses it so there are a lot of resources available to help me. It could be a very easy way to sketch ideas and segment ideas for very simple feedback, development and isolation of bugs.

Without js it cannot be anywhere near as complex or interesting as I would like it to be. And I have not used it in a long time so I would have to relearn most of it.

Final language

As I and many other know python so well I am going to use python, its downsides are far too high level for me to comprehend let alone consider, where the upsides are perfect for the briefs specification of a text based game.

Components

Functions

|  |  |  |
| --- | --- | --- |
| name | input | output |
| Print | String, end | Prints the string to the console and then prints end |
| For | A range to go through | It goes through the range and changes a variable predictably |
| While | Bool / statement | Repeats code inside until bool/statement == false |
| Def | String, parameters and code | Makes the string into a function that runs the given code |
| If | Bool/statement | Runs the code if bool/statement evaluates to true |
| Elif | Bool/statement | Runs the code if bool/statement evaluates to true and no (el)if statement has run before it |
| else |  | Runs the code if no (el)if statement has run before it |
| In | Variables | Checks if the first variable is contained in the second and returns true if so |
| Exec | String | Runs the string as if it were code |
| Format | String | Puts the string into the string this function is applied to |
| Sleep | Integer | Stops the code for the integer amount of seconds |
| Type | Variable | Returns the type of variable it is |
| Input | String | Requests the user for input when presented with the given string returns that input |
| Lower | String | Makes all letters lowercase |
| = | Variable(s) piece(s) of data | Redefines the variables with the pieces of data |
| == | Variable(s) and/or piece(s) of data | Compares the given vales to see if they are the same and returns true if so |
| != | Variable(s) and/or piece(s) of data | Compares the given vales to see if they are the same and returns false if so |
| < | Variable(s) and/or piece(s) of data | Compares the given values to see if the first is greater than the other if so it returns false |
| > | Variable(s) and/or piece(s) of data | Compares the given values to see if the first is greater than the other if so it returns true |
| <= | Variable(s) and/or piece(s) of data | Compares the given values to see if the first is less than or = to the other if so it returns true |
| >= | Variable(s) and/or piece(s) of data | Compares the given values to see if the first is greater than or = to the other if so it returns true |

Data types

|  |  |
| --- | --- |
| Dict | It stores data with a string assigned to it |
| List | It stores data with a predictable order and numbers assigned to it |
| 2d, 3d etc list | A list with multiple lists inside the list. (and potentially multiple lists inside those lists etc.) |
| Library | It stores functions in a .py file |
| .txt | It stores a long string |
| variables | It stores a single piece of data |

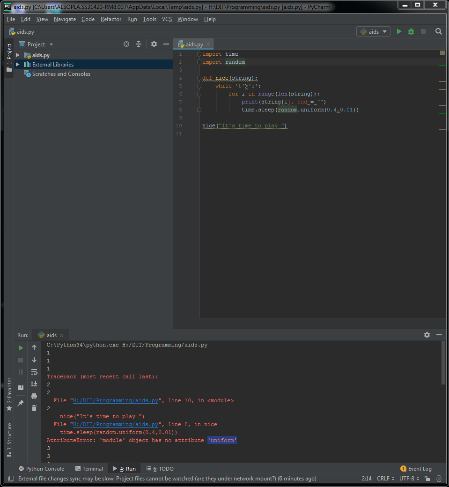
Tools

PyCharm

Looking at most internet reviews, they don’t really have anything functionally bad to say. It’s auto completion is great, it’s file browsing and navigation is superior to everything else, it GUI is great and any modules that are not on your system that you wish to import it can do it for you.

The only downside I can find is that it is slow, it depends on the computer mostly so I will attempt to install it on the school computers(or possibly onto a USB to put into the school computers) and see if it runs as badly as some people say.

After installing it on the school computers.



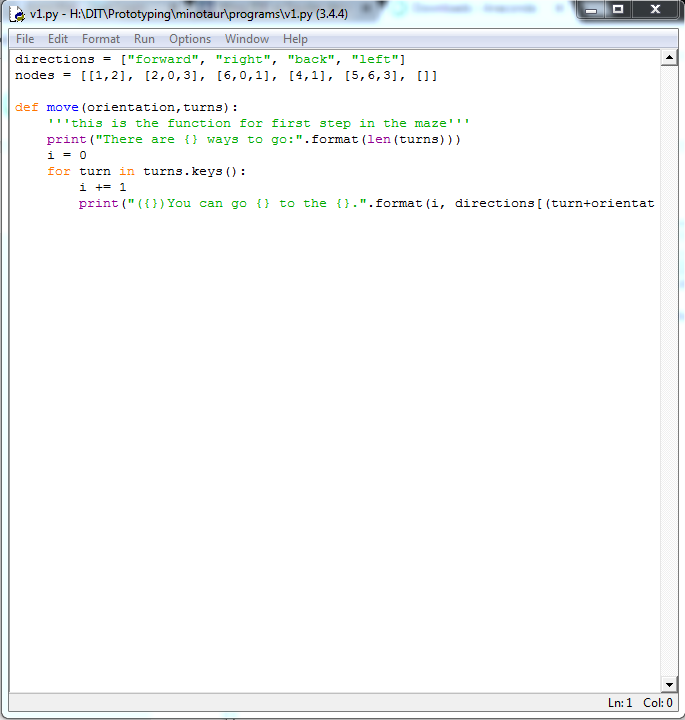
There were some issues. After trying to simply import native libraries from python I had some trouble. I couldn’t import the random library and when I tried to run the code it didn’t print anything but weird numbers that I have no idea what it means.

While I’m sure I could learn what it does and why it is doing it I just don’t have the time to learn how to use the IDE.

IDLE

After using IDLE on the school computers and at home for a while I know that it is well integrated with python with an easy console. The colours are easy to recognise and use, partly because I have already been using them and partly because they are well designed. The main reason why I would want to use IDLE is because I know how to use it and it will not inhibit me in my creative process.

File navigation is clunky at best and debugging can be a pain with no line numbering.



Wing IDE (101/personal)

Wing IDE is said to have a good debugger, easy shortcut programmability, macro capabilities and many plug-ins.

But it is at a price, $39 per user and so even if it was a brilliant piece of software, a price wall immediately removes the possibility of me using it as I haven’t even tried it and I don’t have time to evaluate this purchase let alone learn how to use it.

So I will look at its little brothers, wing 101 and wing personal. After looking on the internet I could not find any in depth reviews on wing 101 and personal and as I could not install it as a user on the school computers I cannot review them for myself.

Spyder

Spyder uses a powerful library for auto completion called rope. It is free which a must for this project as. It can plot graphs but this will not really be useful for the user but could be useful for path analysis in the maze and easily viewable data on pathways and potential changes to the pathways depending on the stakeholder’s feedback. It works on Linux, Mac and Windows. It is lightweight which is good for our relatively slow school computers. And it is solely designed for python which means it is optimised solely for python.

It is ugly, and it is not as good at debugging as other IDEs.

After installing it and using it I really like it, there was no issue with importing libraries, the biggest downside, the debugging is great (better than IDLE the only other IDE that was working for me) as it has line numbers on the side and you can run it in the program itself.

One downside was that the time.sleep function is not as good as IDLE but that could be because I am running it natively in the program and not in the separate console, and even if that is an issue I can run it in IDLE and code it in Spyder. The one last test is if I restart it and it stays on the school computer as every time it restarts it resets the data and if the program is not still on the computer I will not be willing to reinstall it 3 times a week.

Final IDE

Sadly I have to choose IDLE for my prototyping IDE, this is because while it is not the best IDE for the job it is the only one that can be used efficiently on the school computers and any IDE that is online is no good either as my internet is shut down at 9:00pm (that is why I have not looked at any online IDEs).

VCS

Github

More things!

Equipment

I am looking to use a keyboard, mouse and monitor to interact with the program, while it will be mostly keyboard and monitor the mouse will be used to navigate to the program and start it.

While I could use a tablet and phone and I could potentially look into mobile options that will be dependent on if I can find a python program that can import files and of course, stakeholder feedback. When it comes to the prototype there is no question that keyboard and mouse will be far more efficient than the touchscreen keyboards available.

Stakeholders

* Mr Ny
* Other students
* Family

Health and Safety.

To remain healthy and safe during the creation of this prototype there are certain actions that must be taken.

* I will sit up straight to prevent back injuries form prolonged desk work
  + This involves my head being directly above my neck base to minimise the stress on the neck
  + Making full use of my seat and its lumbar support
  + Moving my chair in as far as possible to support my arms with the desk
  + Ensuring the monitor is directly in front of me to not twist my neck unessecerily
* I will have the seat height so that my elbows are at 90° and ensuring that my hands are equally distant from their corresponding shoulder
* I will avoid looking at a monitor for more than an hour and sitting at a desk for more than an hour and if I am planning to do more than an hour session, plan for regular breaks with water and food when needed
  + Rested eyes function better [[1]](#footnote-1)
  + A hydrated brain functions better[[2]](#footnote-2)

Prototyping Techniques/Processes

Story-boarding

Story-boarding is a visual representation of how events will occur in your prototype.

It consists of simple pictures that show what will happen, when and where it will happen and how it will happen.

While for most other games I would really like story boarding as this is a technique used it movies and I think video games should have just as much thought in the story as movies do, but in this case we have a text based video game. So while I could use the technique I don’t think it would be applicable as text is more open to interpretation by the reader and so trying to paint a picture could result in the wrong picture being painted.

Rapid Prototyping

Rapid prototyping consists of three simple stages, prototyping, review, refinement and repeat.

At first a prototype is made this can be a simple prototype or a complex one, it can be a piece of paper or a 50GB program.

Then the prototype is reviewed by the stakeholders, they share what they like, what they don’t and any new ideas they have for the product/prototype.

Then using the feedback from the review the prototype is refined (or possibly scraped and rebuilt) and then the cycle repeats.

As this is the foundation of the entire idea of prototyping it is something I will use in practically every single iteration and for practically every single decision I make after the initial idea.

Wizard of Oz Prototyping

Wizard of Oz prototyping is when you use an analogue program to represent a digital one. For example if it was a website then there would be multiple real pages of paper to represent the possible webpages, usually this is done for visual programs where it is very expensive to make complex visuals compared to the cost of the pens and paper to create them.

I like this idea but again it is for visual products and as mine is text based I cannot directly apply this method but I want to use a simplified version of this, instead of making pages with the actual text. Walk them through my program with my words. This reduces the at least 50 paragraphs I would need to write and the 50ish pages I would need to make.

Agile

Agile is a form of laksjdgfkjashdfpkujagweokfjha;wkj

1. I took the cheap and easy tips from here <https://www.allaboutvision.com/cvs/irritated.htm> [↑](#footnote-ref-1)
2. <https://www.psychologytoday.com/intl/blog/you-illuminated/201010/why-your-brain-needs-water> [↑](#footnote-ref-2)