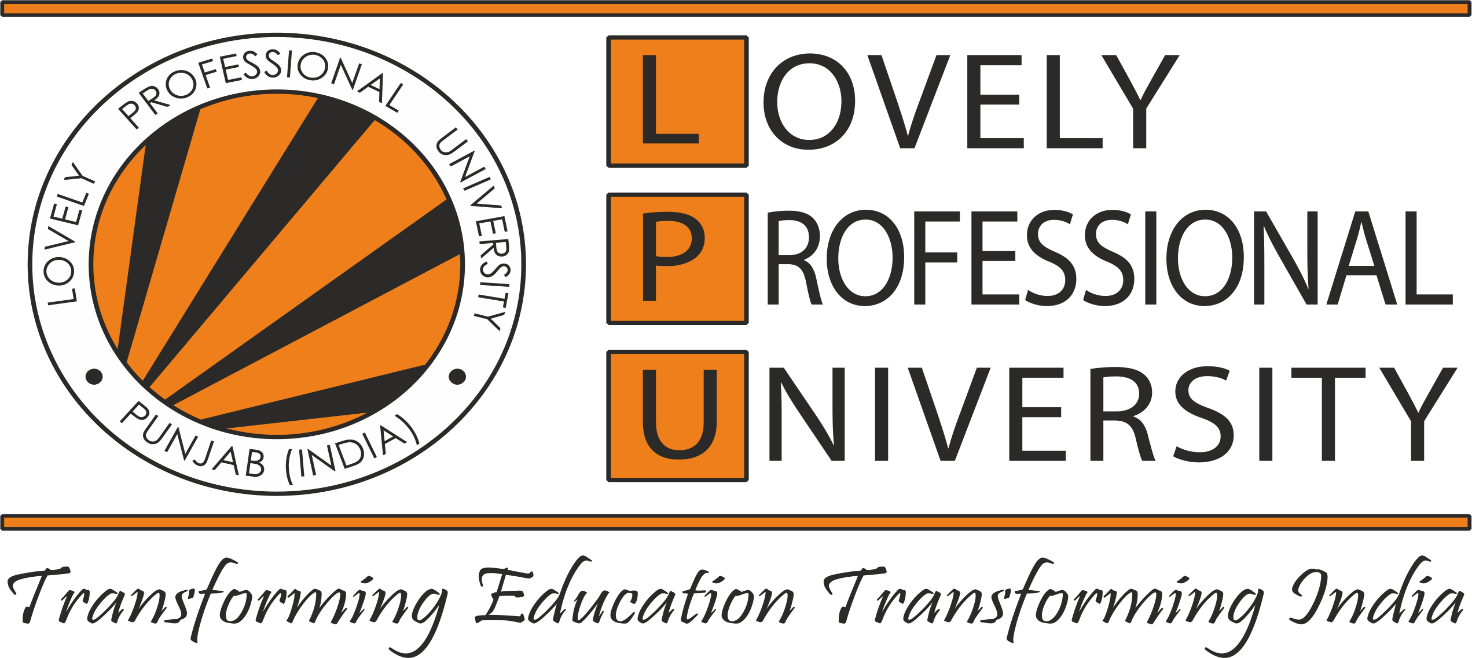
**MINESWEEPER USING PYTHON**

**END TERM REPORT**

** By-**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.no.** | **Name** | **Section** | **Roll no.** |
| **1** | **Ashul yadav** | **K19PG** | **48** |
| **2** | **Aditi Sharma** | **K19PG** | **49** |
| **3** | **Vikas yadav** | **K19PG** | **14** |

**Department of Intelligent Systems,**

**School of Computer Science Engineering,**

**Lovely Professional University, Jalandhar, Punjab**

**November, 2020**

**Student declaration**

**This is to declare that this report has been written by us. No part of the report is copied from other sources. All information included from other sources have been duly acknowledged. We aver that if any part of the report is found to be copied, we are shall take full responsibility for it.**

**Name: Ashul Yadav**

**Aditi Sharma**

**Vikas Yadav**

**Roll no.: 48,49,14**

**Place- Lovely Professional University, jalandhar**

**Date- 28-10-2020**

**Content**

**Title page no.**

1. **Introduction to the project……………………………………… 04**
2. **Objective of the project…………………………………………… 05**
3. **Features………………………………………………………………….. 06**
4. **System and software required………………………………… 06**
5. **Description of the project……………………………………….. 07**

* 5.1 basic grid of minesweeper
* 5.2 input system
* 5.3 data storage
* 5.4 setting up the mines
* 5.6 setting up the grid numbers
* 5.7 game loop
* 5.8 handling the player input
* 5.9 sanitizing the input
* 5.10 handling the flag input
* 5.11 handling the standard input
* 5.12 end game
* 5.13 clearing the console

**6.working of the project……….……………………………………. 12**

**7.work distribution…………………………………………………….. 15**

**8.techonology used…………………………………………………….. 15**

**9.S.W.O.T analysis………………………………………………………. 16**

**Bonafide Certificate**

Certified that this project report “MINESWEEPER USING PYTHON” is the bonafide work

Of “ASHUL YADAV, ADITI, VIKAS YADAV” who carried out the project work under my

supervision.

Dr. Dhanpratap Singh

(Assistant professor in intelligent system) . I.D.25705

Department- School of Computer Science and Engineering

1. **Introduction**

Microsoft Minesweeper (formerly just Minesweeper, and also known as Flower Field) is a [minesweeper](https://en.wikipedia.org/wiki/Minesweeper_(video_game))-type [video game](https://en.wikipedia.org/wiki/Video_game) created by Curt Johnson, originally for [IBM](https://en.wikipedia.org/wiki/IBM)'s [OS/2](https://en.wikipedia.org/wiki/OS/2), that was ported to [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows) by Robert Donner, both [Microsoft](https://en.wikipedia.org/wiki/Microsoft) employees at the time. First officially released as part of the [Microsoft Entertainment Pack 1](https://en.wikipedia.org/wiki/Microsoft_Entertainment_Pack#Microsoft_Entertainment_Pack_1) in 1990, it was first included in the standard install of [Windows 3.1](https://en.wikipedia.org/wiki/Windows_3.1) in 1992, replacing [reversi](https://en.wikipedia.org/wiki/Reversi) from [Windows 3.0](https://en.wikipedia.org/wiki/Windows_3.0).[[1]](https://en.wikipedia.org/wiki/Microsoft_Minesweeper#cite_note-gamesetwatch.com-1) Microsoft Minesweeper was included without major changes in all subsequent Windows releases until [Windows Vista](https://en.wikipedia.org/wiki/Windows_Vista), at which time an updated version by [Oberon Media](https://en.wikipedia.org/wiki/Oberon_Media) replaced it.[[2]](https://en.wikipedia.org/wiki/Microsoft_Minesweeper#cite_note-2) In [Windows 8](https://en.wikipedia.org/wiki/Windows_8) and later the game is not included with a fresh Windows install, but [Microsoft Studios](https://en.wikipedia.org/wiki/Microsoft_Studios) has published an updated version of it, developed by [Arkadium](https://en.wikipedia.org/wiki/Arkadium), on [Windows Store](https://en.wikipedia.org/wiki/Windows_Store). The goal of Minesweeper is to uncover all the squares on a grid that do not contain [mines](https://en.wikipedia.org/wiki/Land_mine) without being "blown up" by clicking on a square with a mine underneath. The location of most mines is discovered through a logical process, but some require guessing, usually with a 50-50 chance of being correct. Clicking on the game board will reveal what is hidden underneath the chosen square or squares (a large number of blank squares [bordering 0 mines] may be revealed in one go if they are adjacent to each other). Some squares are blank while others contain numbers (from 1 to 8), with each number being the number of mines adjacent to the uncovered square.

To help the player avoid hitting a mine, the location of a suspected mine can be marked by flagging it with the right [mouse button](https://en.wikipedia.org/wiki/Mouse_button). The game is won once all blank or numbered squares have been uncovered by the player without hitting a mine; any remaining mines not identified by flags are automatically flagged by the computer. However, in the event that a game is lost and the player had mistakenly flagged a safe square, that square will either appear with a red X, or else a red X covering the mine (both denoting the square as safe). The game board comes in three set sizes with a predetermined number of mines: "beginner", "intermediate", and "expert", although a "custom" option is available as well.

There are many more new versions of minesweeper added to the platform with advanced features and are easy to play with easy algorithm and more fun.

Minesweeper genius is one of the latest version of minesweeper added to the list.

**Minesweeper Genius** reboots the simple little game we all played ages ago and expands on it with elements reminiscent of Sudoku and maze-runner games like Bomber man. The idea is familiar – use logic to avoid deadly mines on a grid – but this time around we have a hero and a bit of a backstory. You play as Aristotle, the adorable genius, as he literally sweeps a path through the grid to the exit using logic and special moves to avoid the deadly mines. Aliens are to blame for Aristotle’s plight, as part of some mysterious extra-terrestrial science experiments, but using your brain to navigate through minefields is what it’s really all about.

**Minesweeper Genius** lets you play in campaign mode or custom mode. In campaign mode, you progress through randomly generated maps to unlock more levels and abilities. The advancement factor is one of the game’s biggest improvements over the original Minesweeper, and it provides a satisfying path to overcoming challenges and achieving goals. As you move up the campaign ladder, you’ll earn the ability to flag suspected mines and you’ll discover special tiles that let Aristotle jump over adjacent tiles or rearrange the map. Higher levels also add new puzzle-solving elements, like indicating not only how many mines are adjacent to a given tile, but also how many are in each row and column.

**2. Objective of the project**

Python is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language), [high-level](https://en.wikipedia.org/wiki/High-level_programming_language) and [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language). Created by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) and first released in 1991, Python's design philosophy emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability) with its notable use of [significant whitespace](https://en.wikipedia.org/wiki/Off-side_rule). Its [language constructs](https://en.wikipedia.org/wiki/Language_construct) and [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) approach aim to help [programmers](https://en.wikipedia.org/wiki/Programmers) write clear, logical code for small and large-scale projects.

Python is [dynamically typed](https://en.wikipedia.org/wiki/Dynamic_programming_language) and [garbage-collected](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigms), including [structured](https://en.wikipedia.org/wiki/Structured_programming) (particularly, [procedural](https://en.wikipedia.org/wiki/Procedural_programming)), [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), and [functional programming](https://en.wikipedia.org/wiki/Functional_programming). Python is often described as a "batteries included" language due to its comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).

Python was created in the late 1980s as a successor to the [ABC language](https://en.wikipedia.org/wiki/ABC_(programming_language)). Python 2.0, released in 2000, introduced features like [list comprehensions](https://en.wikipedia.org/wiki/List_comprehension) and a garbage collection system with [reference counting](https://en.wikipedia.org/wiki/Reference_counting).

Python 3.0, released in 2008, was a major revision of the language that is not completely [backward-compatible](https://en.wikipedia.org/wiki/Backward_compatibility), and much Python 2 code does not run unmodified on Python 3.

This project include the development of the game (minesweeper) using python language, making us more familiar to the language and helping us to understand the basic as well as advanced uses of this language. This helped us to put up our theory into practical and coming up with such an interesting project that boosts our confidence as well as provides us confidence to work more easily and in a better way in future.

This project implements our knowledge and creative skills with algorithms and various other fields and brings in together to form the game. From loops to memory our learned knowledge is implemented in this.

Particularly the objectives are-

* Creating a game (minesweeper) using python
* Implementing the use of various algorithms required
* Uses of loops and memory for implementation of the further process of game.
* Using a simple and easy GUI (Graphical User Interface)
* Using algorithms and various other tools available for the betterment of the game
* Developing various skills like teamwork and problem-solving skills
* Helping students understand the working concept of software development as well as game development
* Making students familiar with IDE
* Making them understand the concepts of python more clearly.
* Developing the habit and improving their coding skills

Overall, this project develops the coding skills as well as implementation of knowledge in the practical world for a student. Helps them understand the various ways to develop and move further with python and implementation of algorithm with connection to various other features such as memory allocation and database management to remember the exact place where a person left the game. Not only python but it develops the overall personality by improving the teamwork capability of a student in accordance with leadership skills.

**3. Features**

**3.1- algorithmics approach**

Early algorithms developed to solve Minesweeper focus on the deterministic deductions needed to uncover safe moves. Adamatzky modelled Minesweeper as a cellular automaton. The cell state is given two components. The first component indicates whether the cell is either covered, uncovered, or a mine. The second component, which is only available to the system if the cell is uncovered, is the number of mines adjacent to that particular cell. One limitation to the CA model is in the transition function which only accounts for two basic deductions in Minesweeper. However, its main weakness is its tendency to become stuck on particular configurations. Since the CA solver is deterministic, it never makes guesses or random moves, an occasionally necessary step in completing a game especially at harder difficulties. Adamatzky expressed the need for stochastic features necessary for future algorithms.

**3.2- simple GUI**

The graphical user interface is a form of [user interface](https://en.wikipedia.org/wiki/User_interface) that allows [users](https://en.wikipedia.org/wiki/User_(computing)) to [interact with electronic devices](https://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction) through graphical [icons](https://en.wikipedia.org/wiki/Icon_(computing)) and audio indicator such as primary notation, instead of [text-based user interfaces](https://en.wikipedia.org/wiki/Text-based_user_interface), typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep [learning curve](https://en.wikipedia.org/wiki/Learning_curve) of [command-line interfaces](https://en.wikipedia.org/wiki/Command-line_interface) (CLIs), which require commands to be typed on a [computer keyboard](https://en.wikipedia.org/wiki/Computer_keyboard).

The actions in a GUI are usually performed through [direct manipulation](https://en.wikipedia.org/wiki/Direct_manipulation_interface) of the graphical elements.

**4. System and software required**

The basic software and system requirements for this project is as follows-

**Minimum requirements (system) –**

* Modern operating system:

Windows 7 or 10

Linux: RHEL 6/7, 64 bits

Mac OS X 10.11 or higher, 64-bit

* X 86 64-bit CPU (intel i3 and above /AMD architecture)
* 4 GB RAM
* Python version 2.7 or above

**Recommended requirements(system) –**

* Modern operating system:

Windows 7 or 10

Linux: RHEL 6/7, 64 bits

Mac OS X 10.11 or higher, 64-bit

* X 86 64-bit CPU (intel i5 and above /AMD architecture)
* 8 GB RAM
* Python version 3.0

**Software requirements –**

* Python version 3.0 and above
* IDE (Spyder, Anaconda, PyCharm etc.)

**5. Description of the project**

**5.1 basic grid of minesweeper –**

Before creating the game logic, we need to design the basic layout of the game. A square grid is rather easy to create using Python. We will be using function and keywords such as global and loops for grid formation.

**Keyword used- global**

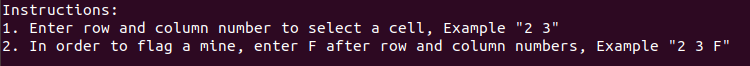
global keyword allows you to modify the variable outside of the current scope. It is used to create a global variable and make changes to the variable in a local context.

The basic rules for global keyword in Python are:

* When we create a variable inside a function, it is local by default.
* When we define a variable outside of a function, it is global by default. You don't have to use global keyword.
* We use global keyword to read and write a global variable inside a function.
* Use of global keyword outside a function has no effect.

**5.2 Input system**

In our version of Minesweeper, we will be using the row and column numbers for our input technique. Before starting the game, the script must provide a set of instructions for the player.



The row and column numbers displayed along with the grid are helpful for our input system. As we know, keeping track of mines without any indicator can be difficult. Therefore, Minesweeper has a provision of using ‘flag’ to mark the cells, which we know contains a mine.

**5.3 Data storage**

For a single game of Minesweeper, we need to keep track of the following information:

* The **size** of the grid.
* The **number of mines**.
* **The ‘actual’ grid values** – At the start of the game, we need a container for storing the real values for the game, unknown to the player. For instance, the location of mines.
* **The ‘apparent’ grid values** – After each move, we need to update all the values that must be shown to the player.
* **The flagged positions** – The cells which have been flagged.

Data structure is used in this structure of code. Values are stored and are retrieved at specific time with randomness.

**Variable used- \_\_name\_\_**

The \_\_name\_\_ variable is a special Python variable. It gets its value depending on how we execute the containing script.

Sometimes we write a script with functions that might be useful in other scripts as well. In Python, we can import that script as a module in another script.

Thanks to this special variable, we can decide whether we want to run the script. Or that we want to import the functions defined in the script.

**5.4 Setting up the mines**

We need to set up the positions of the mines randomly, so that the player might not predict their positions. In the code, we choose a random number from all possible cells in the grid. We keep doing this until we get the said number of mines.

**Function used- ‘randint’**

The randint() method returns an integer number selected element from the specified range. Note: This method is an alias for randrange(start, stop+1) . This function helps us to generate random no. of grid locations to place mines every time the game is executed.

**5.5 setting up the grid numbers**

For each cell in the grid, we have to check all adjacent neighbours whether there is a mine present or not. Loops are used to compare the adjacent values for any concurrency of values. These values are to be hidden from the player, therefore they are stored in number variable.

**5.6 game loop**

Game Loop is a very crucial part of the game. It is needed to update every move of the player as well as the conclusion of the game. In each iteration of the loop, the Minesweeper grid must be displayed as well as the player’s move must be handled.

**5.7 handling the player input**

There are 2 types of input-

* **Standard input**

In a normal kind of move, the row and column number are mentioned. The player’s motive behind this move is to unlock a cell that does not contain a mine.

* **Flag input**

In a flagging move, three values are sent in by the gamer. The first two values denote cell location, while the last one denotes flagging.

**5.8 sanitize the input**

After storing the input, we have to do some sanity checks, for the smooth functioning of the game.

**Function used- continue**

Continue is also a loop control statement just like the break statement. continue statement is opposite to that of break statement, instead of terminating the loop, it forces to execute the next iteration of the loop.  
As the name suggests the continue statement forces the loop to continue or execute the next iteration. When the continue statement is executed in the loop, the code inside the loop following the continue statement will be skipped and the next iteration of the loop will begin.

**5.9 handling the flag input**

Managing the flag input is not a big issue. It requires checking for some pre-requisites before flagging the cell for a mine.

The following checks must be made:

* The cell has already been flagged or not.
* Whether the cell to be flagged is already displayed to the player.
* The number of flags does not exceed the number of mines.
* After taking care of these issues, the cell is flagged for a mine.

**5.10 Handling the standard input**

The standard input involves the overall functioning of the game. There are three different scenarios:

* **Anchoring on a mine**

After we land on a cell with mine, we need to display all the mines in the game and alter the variable behind the game loop.

* **Visiting the ‘0’ valued cell**

The trickiest part of creating the game is managing this scenario. Whenever a gamer, visits a ‘0’-valued cell, all the neighbouring elements must be displayed until a non-zero-valued cell is reached. This objective is achieved using [Recursion](https://www.askpython.com/python/python-recursion-function). Recursion is a programming tool in which the function calls itself until the base case is satisfied. The neighbours function is a recursive one, solving our problem.

* **Choosing a non-zero valued cell**

No effort is needed to handle this case, as all we need to do is alter the displaying value.

**5.11 End game**

There is a requirement to check for completion of the game, each time a move is made. The function check\_over(). is responsible for checking the completion of the game. We count the number of cells, that are not empty or flagged. When this count is equal to the total cells, except those containing mines, then the game is regarded as over.

**5.12 Clearing output each move**

The terminal becomes crowded as we keep on printing stuff on it. Therefore, there must be provision for clearing it constantly.

**Note:** There is a need to import the os library, before using this feature. It can be done by ‘import os’ at the start of the program.

**Working of the project**

**Calendar

Description automatically generated**

When we run the program, a similar screen pops up in the console showing us the grid of the minesweeper. It asks the player to input the row and column no. of the first grid which he wants to reveal.

Let us assume we enter 3,4 so the grid at 3,4 will be revealed. If it will be a mine then it will show the result as game over you landed on a mine, and if we didn’t land on the mine then the program will ask user to enter the no. of row and column once again to reveal another grid cell until we land on a mine.

**Calendar

Description automatically generated**

In this image we can see that we landed on a mine and the output shows “landed on a mine. GAME OVER !!!!”. if we look closely we can see a output at 1,2 cell this is due to the user who entered the value as 1,2 before the final input which led to triggering a mine.

Calendar

Description automatically generated

We have used ‘F’ keyword as a particular sign to depict the cell which is flagged. We execute this by adding a keyword ‘F’ after entering the cell no. to be revealed. If we look closely we can observe this in the input.

**Work distribution**

The amount of work required in this project to be done was divided among the team members equally.

This team consisted of 3 members- Ashul yadav, Aditi, Vikas yadav

We all together designed this project and made best out of our knowledge to complete this project.

Development of grid and various other inputs was done by Vikas making use of all available libraries and functions such as randint, clear() etc.

And Aditi handled the output and its analysis including sanitization of output and handling the flag output with standard output making use of libraries such as global, os, etc.

And I have developed the remaining of the program which included handling the cases and placement of cells in the grid random placements of mines using functions such as \_\_name\_\_, random etc.

And the remaining loops of the game were developed by all the members together which included data storage, clearing the output and other issues that needed to be solved.

**Technology used**

The required technology for this project includes the types of modules required and type of functions with number of keywords and techniques used**.**

**Function used- continue**

Continue is also a loop control statement just like the break statement. continue statement is opposite to that of break statement, instead of terminating the loop, it forces to execute the next iteration of the loop.  
As the name suggests the continue statement forces the loop to continue or execute the next iteration. When the continue statement is executed in the loop, the code inside the loop following the continue statement will be skipped and the next iteration of the loop will begin.

**Function used- ‘randint’**

The randint() method returns an integer number selected element from the specified range. Note: This method is an alias for randrange(start, stop+1). This function helps us to generate random no. of grid locations to place mines every time the game is executed.

**Variable used- \_\_name\_\_**

The \_\_name\_\_ variable is a special Python variable. It gets its value depending on how we execute the containing script.

Sometimes we write a script with functions that might be useful in other scripts as well. In Python, we can import that script as a module in another script.

Thanks to this special variable, we can decide whether we want to run the script. Or that we want to import the functions defined in the script.

**Module used- random**

This module implements pseudo-random number generators for various distributions.

For integers, there is uniform selection from a range. For sequences, there is uniform selection of a random element, a function to generate a random permutation of a list in-place, and a function for random sampling without replacement.

On the real line, there are functions to compute uniform, normal (Gaussian), lognormal, negative exponential, gamma, and beta distributions. For generating distributions of angles, the von Mises distribution is available.

**Module used- OS**

This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see [open()](https://docs.python.org/3/library/functions.html#open), if you want to manipulate paths, see the [os.path](https://docs.python.org/3/library/os.path.html#module-os.path) module, and if you want to read all the lines in all the files on the command line see the [fileinput](https://docs.python.org/3/library/fileinput.html#module-fileinput) module. For creating temporary files and directories see the [tempfile](https://docs.python.org/3/library/tempfile.html#module-tempfile) module, and for high-level file and directory handling see the [shutil](https://docs.python.org/3/library/shutil.html#module-shutil) module.

**SWOT analysis**

**Strength-**

It includes complex algorithm for the mines to be placed and thus it makes it difficult for the player to crack this leading him to makes his thinking capacity and algorithm to be good and fast enough to crack the cells with mines. It strengthens the mental ability as well as practical knowledge of the player.

**Weakness-**

weakness include the dependency of execution of program on algorithm. Its difficult for the player to reach that algorithm. Although there are ways to solve minesweeper, but a general person cannot do so, it requires algorithmic knowledge.

**Opportunities –**

This project provides ample of opportunities to a person to develop his coding skills as well as algorithmic approach toward a particular task. It helps the programmer to improve his skills in critical thinking as well as teamwork.

**Threats-**

As according to us there are no threats found in this project.