QUARANTINE SNAKE



**UNDER THE SUPERVISION OF**

# Guide

# 

**ARUN MITTAL**

**SUBMITTED BY**

**NIRAJ GUPTA (190496)**

**GREATR NOIDA INSTITUTE OF TECHNOLOGY**

Approved by AICTE and Affiliated to

AKTU Lucknow,

**DEPARTNMENT OF COMPUTER SCIENCE**

**INDEX**

1. INTRODUCTION
2. DISCRIPTION
3. SYSTEM ANALYSIS
4. FEASIBILITY REPORT
5. SYSTEM DESIGN
6. SCREEN OUTPUT
7. CONCLUSION
8. BIBLIOGRAPHY/REFERENCES

**DECLARATION**

I, hereby declare that, this project entitled Quarantine Snake is the

Bonafede work of mine carried out under the supervision of

Mr. Arun Mittal, I declare that, to the best of my knowledge,

the work reported herein does not form part of any other project report or

dissertation on the basis of which a degree or award was conferred on an earlier

occasion to any other candidate. The content of this report is not being presented

by any other student to this or any other University for the award of a degree.

**Date:**

Student’s Name: NIRAJ GUPTA

Student ID: 190496

**SUBMITTED TO:**

**Mr. Arun Mittal   
Guide**

**CERTIFICATE**



**ACKNOWLEDGEMENT**

I have taken efforts in this project. however, it would not have been possible without the kind support and help of many individuals and organization. I would like to extend my sincere thanks to all of them.

I am highly indebted to Mr. Arun Mittal, Professor, Department of

Computer Science), for his excellent guidance, positive criticism and valuable comments.

**INTRODUCTION**

***Snake*** is the common name for a video game concept where the player maneuvers a line which grows in length, with the line itself being a primary obstacle. The concept originated in the 1976 arcade game *Blockade*, and the ease of implementing *Snake* has led to hundreds of versions (some of which have the word *snake* or *worm* in the title) for many platforms. After a variant was preloaded on Nokia mobile phones in 1998, there was a resurgence of interest in the snake concept as it found a larger audience. There are over 300 *Snake*-like games for ios alone

The player controls a dot, square, or objects on a bordered plane. As it moves forward, it leaves a trail behind, resembling a moving snake. In some games, the end of the trail is in a fixed position, so the snake continually gets longer as it moves. In another common scheme, the snake has a specific length, so there is a moving tail a fixed number of units away from the head. The player loses when the snake runs into the screen border, a trail or other obstacle, or itself.

The Snake concept comes in two major variants:

1. In the first, which is most often a two-player game, there are multiple snakes on the playfield. Each player attempts to block the other so he or she runs into an existing trail and loses. *Surround* for the Atari 2600 is an example of this type. The Light Cycles segment of the *Torn* arcade game is a single-player version where the other "snakes" are AI controlled.
2. In the second variant, a sole player attempts to eat items by running into them with the head of the snake. Each item eaten makes the snake longer, so controlling is progressively more difficult. Examples: *Nibbler*, *Snake Bite*.



**DISCRIPTION**

The *Snake* design dates back to the arcade game *Blockade*, developed and published by Gremlin in 1976. It was cloned as *Bigfoot Bonkers* the same year. In 1977, Atari released two *Blockade*-inspired titles: the arcade game *Dominos* and Atari VCS game *Surround* Surround was one of the nine Atari VCS (later the Atari 2600) launch titles in the United States and was also sold by Sears under the name *Chase*. That same year, a similar game was launched for the Bally Astrocade as *Checkmate*.

The first known personal computer version, titled *Worm*, was programmed in 1978 by Peter Trefonas of the US on the TRS-80, and published by *CLOAD* magazine in the same year. This was followed shortly afterwards with versions from the same author for the Commodore PET and Apple II. A microcomputer clone of the *Hustle* arcade game, itself a clone of *Blockade*, was written by Peter Trefonas in 1979 and published by *CLOAD*. An authorized version of *Hustle* was published by Milton Bradley for the TI-99/4A in 1980. In 1982's *Snake* for the BBC Micro, by Dave Breanne, the snake is controlled using the left and right arrow keys relative to the direction it is heading in. The snake increases in speed as it gets longer, and there's only one life; one mistake means starting from the beginning.

*Nibbler* (1982) is a single-player arcade game where the snake fits tightly into a maze, and the gameplay is faster than most snake designs. Another single-player version is part of the 1982 *Torn* arcade game, themed with light cycles. It reinvigorated the snake concept, and many subsequent games borrowed the light cycle theme.

Starting in 1991, *Nibbles* was included with MS-DOS for a period of time as a QBasi sample program. In 1992, *Rattler Race* was released as part of the second *Microsoft Entertainment Pack*. It adds enemy snakes to the familiar apple-eating gameplay.

*Meerca Chase* is a snake game available on Neopost.

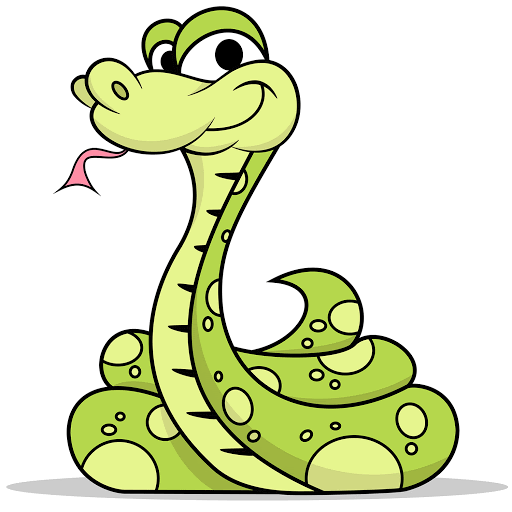
*Slither.io* (2016) is a popular multiplayer interpretation of *Snake.*

In 2017, Google released their version of the game as an Easter egg, whenever the phrases "snake", "plays snake”, “snake game" and "snake video game" are typed.

This simple Snake Game project is written in Python. The project file contains image files and python scripts (game.py, duplicate.py). GUI uses pygame library. Talking about the gameplay, it’s a single player game, where the player (Snake) has to eat all the apples in order to grow longer. The main objective of this game is to gain the highest score. While playing the game, make sure the snake should not cross the edges otherwise you’ll die.

A simple and clean GUI is provided for easy gameplay. The gameplay design is so simple that the user won’t find it difficult to use and understand. Different images are used in the development of this game project. In order to run the project, you must have installed Python and Pygame on your PC. This simple game project with source code is free to download. Use for education purpose only! For the project demo,

This project creates the Snake Game seen in the Worked Example for the 9/10 MOOC, using a Makey Makey controller to build skill in the use of digital systems, Object-Oriented programming design and implementation, and data representation. Students will learn about Object-Oriented programming using existing Classes and Objects, as well in designing and creating multiple new Classes, making this a good intermediate project. An understanding of general and Object-Oriented programming concepts. Digital Technologies Summary This activity introduces students to: complex digital systems, including the use of MakeyMakey controller and transmission in networked environments. Object-Oriented design and implementation, including the analysis, tracing and understanding and existing Object-Oriented software components and the design and creation of new Object-Oriented software components. The lesson introduces the students to the idea of designing interactive programs, developing systems thinking, focusing on how different aspects of a program interact with each other.. Band Content Descriptors Investigate the role of hardware and software in managing, controlling and securing the movement of and access to data in networked digital systems Analyse and visualise data to create information and address complex problems, and model processes, entities and their relationships using structured data Design the user experience of a digital system by evaluating alternative designs against criteria including functionality, accessibility, usability, and aesthetics Design algorithms represented diagrammatically and in structured English and validate algorithms and programs through tracing and test cases

 **PYTHON** 

 High-levll, general-purpose **Python** is an interpreted programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

The Python 2 language was officially discontinued in 2020 (first planned for 2015), and "Python 2.7.18 is the last Python 2.7 release and therefore the last Python 2 release. No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.5. and later are supported.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

Python was conceived in the late 1980ss by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language(itself inspired by SETL),capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's *Benevolent Dictator For Life*, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker.He now shares his leadership as a member of a five-person steering council. In January 2019, active Python core developers elected Brett Cannon, Nick Coghlan, Barry Warsaw, Carol Willing and Van Rossum to a five-member "Steering Council" to lead the project.

Python 2.0 was released on 16 October 2000 with many major new features, including a cycle-detecting garbage  and support for Unicode.

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible Many of its major features were backported to Python 2.6.x[ and 2.7.x version series. Releases of Python 3 include the  utility, which automates (at least partially) the translation of Python 2 code to Python 3. Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3.

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by met programming  and met objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

The language's core philosophy is summarized in the document *The Zen of Python* (*PEP 20*), which includes aphorisms such as:

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Readability counts.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum' vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" design philosophy. Alex Martell, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is *not* considered a compliment in the Python culture.

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the CPython reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name—a tribute to the British comedy group Monty Python—and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard foo and bar.

A common neologism in the Python community is *pythonic*, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called *unpythonic*.

Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as *Pythonistas*. *Main article: Python syntax and semantics*

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal.

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is sometimes termed the off-side rule, which some other languages share, but in most languages indentation doesn't have any semantic meaning. Python's statements include (among others):

The assignment statement (token '=', the equals sign). This operates differently than in traditional imperative programming languages, and this fundamental mechanism (including the nature of Python's version of *variables*) illuminates many other features of the language. Assignment in [C](https://en.wikipedia.org/wiki/C_(programming_language)), e.g., x = 2, translates to "typed variable name x receives a copy of numeric value 2". The (right-hand) value is copied into an allocated storage location for which the (left-hand) variable name is the symbolic address. The memory allocated to the variable is large enough (potentially quite large) for the declared type. In the simplest case of Python assignment, using the same example, x = 2, translates to "(generic) name x receives a reference to a separate, dynamically allocated object of numeric (int) type of value 2." This is termed *binding* the name to the object. Since the name's storage location doesn't *contain* the indicated value, it is improper to call it a *variable*. Names may be subsequently rebound at any time to objects of greatly varying types, including strings, procedures, complex objects with data and methods, etc. Successive assignments of a common value to multiple names, e.g., x = 2; y = 2; z = 2 result in allocating storage to (at most) three names and one numeric object, to which all three names are bound. Since a name is a generic reference holder it is unreasonable to associate a fixed data type with it. However at a given time a name will be bound to *some* object, which **will** have a type; thus there is dynamic typing.

The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).

The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.

The while statement, which executes a block of code as long as its condition is true.

The try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.

The raise statement, used to raise a specified exception or re-raise a caught exception.

The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.

The def statement, which defines a function or method.

The with statement, from Python 2.5 released in September 2006, which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing Resource Acquisition Is Initialization (RAII)-like behavior and replaces a common try/finally idiom.

The [break](https://en.wikipedia.org/wiki/Break_statement) statement, exits from the loop.

The continue statement, skips this iteration and continues with the next item.

The pass statement, which serves as a NOP. It is syntactically needed to create an empty code block.

The assert statement, used during debugging to check for conditions that ought to apply.

The yield statement, which returns a value from a generator function. From Python 2.5, yield is also an operator. This form is used to implement coroutines.

The import statement, which is used to import modules whose functions or variables can be used in the current program. There are three ways of using import: import <module name> [as <alias>] or from <module name> import \* or from <module name> import <definition 1> [as <alias 1>], <definition 2> [as <alias 2>], ....

The print statement was changed to the print () function in Python 3.

Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will However, better support for coroutine-like functionality is provided in 2.5, by extending Python's generators. Before 2.5, generators were lazy iterators; information was passed unidirectionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.

ome Python expressions are similar to languages such as [C](https://en.wikipedia.org/wiki/C_(programming_language)) and Java, while some are not:

Addition, subtraction, and multiplication are the same, but the behavior of division differs. There are two types of divisions in Python. They are floor division (or integer division) // and floating point/division. Python also added the \*\* operator for exponentiation.

From Python 3.5, the new @ infix operator was introduced. It is intended to be used by libraries such as NumPy for matrix multiplication

From Python 3.8, the syntax :=, called the 'walrus operator' was introduced. It assigns values to variables as part of a larger expression.

In Python, == compares by value, versus Java, which compares numerics by value[]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-76) and objects by reference.[]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-77) (Value comparisons in Java on objects can be performed with the equals() method.) Python's is operator may be used to compare object identities (comparison by reference). In Python, comparisons may be chained, for example a <= b <= c.

Python uses the words and, or, not for its boolean operators rather than the symbolic &&, ||, ! used in Java and C.

Python has a type of expression termed a *list comprehension*. Python 2.4 extended list comprehensions into a more general expression termed a *generator expression*.

Anonymous functions are implemented using lambda expressions; however, these are limited in that the body can only be one expression.

Conditional expressions in Python are written as x if c else y (different in order of operands from the c ? x : y operator common to many other languages).

Python makes a distinction between lists and tuples. Lists are written as [1, 2, 3], are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be immutable in Python). Tuples are written as (1, 2, 3), are immutable and thus can be used as the keys of dictionaries, provided all elements of the tuple are immutable. The + operator can be used to concatenate two tuples, which does not directly modify their contents, but rather produces a new tuple containing the elements of both provided tuples. Thus, given the variable t initially equal to (1, 2, 3), executing t = t + (4, 5) first evaluates t + (4, 5), which yields (1, 2, 3, 4, 5), which is then assigned back to t, thereby effectively "modifying the contents" of t, while conforming to the immutable nature of tuple objects. Parentheses are optional for tuples in unambiguous contexts.

Python features *sequence unpacking* wherein multiple expressions, each evaluating to anything that can be assigned to (a variable, a writable property, etc.), are associated in the identical manner to that forming tuple literals and, as a whole, are put on the left hand side of the equal sign in an assignment statement. The statement expects an *iterable* object on the right hand side of the equal sign that produces the same number of values as the provided writable expressions when iterated through, and will iterate through it, assigning each of the produced values to the corresponding expression on the left.

Python has a "string format" operator %. This functions analogous to printf format strings in [C](https://en.wikipedia.org/wiki/C_(programming_language)), e.g. "spam=%s eggs=%d" % ("blah", 2) evaluates to "spam=blah eggs=2". In Python 3 and 2.6+, this was supplemented by the format () method of the str class, e.g. "spam={0} eggs={1}".format("blah", 2). Python 3.6 added "f-strings": blah = "blah"; eggs = 2; f'spam={blah} eggs={eggs}'

Python has various kinds of string literals:

Strings delimited by single or double quote marks. Unlike in Unix shells, Perl and Perl-influenced languages, single quote marks and double quote marks function identically. Both kinds of string use the backslash (\) as an escape character. String interpolation became available in Python 3.6 as "formatted string literals

Triple-quoted strings, which begin and end with a series of three single or double quote marks. They may span multiple lines and function like here documents in shells, Perl and Ruby.

Raw string varieties, denoted by prefixing the string literal with an r. Escape sequences are not interpreted; hence raw strings are useful where literal backslashes are common, such as regular expressions and Windows-style paths. Compare "@-quoting" in C#.

Python has array index and array slicing expressions on lists, denoted as a[key], a[start:stop] or a[start:stop:step]. Indexes are zero-based, and negative indexes are relative to the end. Slices take elements from the *start* index up to, but not including, the *stop* index. The third slice parameter, called *step* or *stride*, allows elements to be skipped and reversed. Slice indexes may be omitted, for example a[:] returns a copy of the entire list. Each element of a slice is a shallow copy.

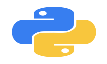
In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as Common Lisp, Scheme, or Ruby. This leads to duplicating some functionality. For example:

List comprehensions vs. for-loops

Conditional expressions vs. if blocks

The eval() vs. exec() built-in functions (in Python 2, exec is a statement); the former is for expressions, the latter is for statements.

Statements cannot be a part of an expression, so list and other comprehensions or lambda expressions, all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as a = 1 cannot form part of the conditional expression of a conditional statement. This has the advantage of avoiding a classic C error of mistaking an assignment operator = for an equality operator == in conditions: if (c = 1) { ... } is syntactically valid (but probably unintended) C code but if c = 1: ... causes a syntax error in Python. Python uses duck typing  and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically typed, Python is strongly typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Python allows programmers to define their own types using classes, which are most often used for object-oriented programming. New instances of classes are constructed by calling the class (for example, SpamClass() or EggsClass()), and the classes are instances of the metaclass type (itself an instance of itself), allowing metaprogramming and reflection. 

**PYGAMES**

**Pygame** is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language. Pygame was originally written by Pete Shinners to replace PySDL after its development stalled. It has been a community project since 2000 and is released under the open sourc free software GNU Lesser General Public License.

Pygame version 2 was planned as "Pygame Reloaded" in 2009, but development and maintenance of pygame completely stopped until the end of 2016 with version 1.9.1. After the release of version 1.9.5 on March 2019, development of new version 2 is active on the roadmap.

* 2019-5-5 pre-release Dev.1
* 2019-6-20 pre-release Dev.2
* 2019-7-14 pre-release Dev.3
* 2019-10-8 pre-release Dev.4
* 2019-10-28 pre-release Dev.6

Pygame uses the Simple DirectMedia Layer (SDL) library, with the intention of allowing real-time computer game development without the low-level mechanics of the C programming language and its derivatives. This is based on the assumption that the most expensive functions inside games can be abstracted from the game logic, making it possible to use a high-level programming language, such as Python, to structure the game.

Other features that SDL doesn't have include vector math, collision detection, 2d sprite scene graph management, MIDI support, camera, pixel-array manipulation, transformations, filtering, advanced freetyp font support, and drawing.

Applications using pygame can run on Android phones and tablets with the use of pygame Subset for Android (pgs4a). Sound, vibration, keyboard, and accelerometer are supported on Android.

There is a regular competition, called PyWeek, to write games using Python (and usually but not necessarily, pygame). The community has created many tutorials for pygame

Pygame is free. Released under the LGPL licence, you can create open source, freeware, shareware, and commercial games with it. See the licence for full details.

For a nice introduction to pygame, examine the line-by-line chimp tutorial, and the introduction for python programmers. buffer, and many other different backends... including an ASCII art backend! OpenGL is often broken on linux systems, and also on windows systems - which is why professional games use multiple backends.

**Multi core CPUs can be used easily**. With dual core CPUs common, and 8 core CPUs cheaply available on desktop systems, making use of multi core CPUs allows you to do more in your game. Selected pygame functions release the dreaded python GIL, which is something you can do from C code.

**Uses optimized C and Assembly code for core functions**. C code is often 10-20 times faster than python code, and assembly code can easily be 100x or more times faster than python code.

**Comes with many Operating systems**. Just an apt-get, emerge, pkg\_add, or yast [this paragraph seems unfinished]

**Uses optimized C and Assembly code for core function** install away. No need to mess with installing it outside of your operating system's package manager. Comes with binary pos system installers (and uninstallers) for Windows or MacOSX. Pygame does not require setup tools with even ctypes to install.

**Truly portable**. Supports Linux (pygame comes with most main stream linux distributions), Windows (95, 98, ME, 2000, XP, Vista, 64-bit Windows, etc), Windows CE, BeOS, MacOS, Mac OS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX, and QNX. The code contains support for AmigaOS, Dreamcast, Atari, AIX, OSF/Tru64, RISC OS, SymbianOS and OS/2, but these are not officially supported. You can use it on hand held devices, game consoles and the One Laptop Per Child (OLPC) computer.

**It's Simple** and easy to use. Kids and adults make shooter games with pygame. Pygame is used in the OLPC project and has been taught in essay courses to young kids and college students. It's also used by people who first programmed in z80 assembler or c64 basic.

**Many games have been published**. Including Indie Game Festival finalists, Australian Game festival finalists, popular shareware, multimedia projects and open source games. Over 660 projects have been published on the pygame websites such as: list needed. Many more games have been released with SDL (which pygame is based on), so you can be sure much of it has been tested well by millions of users.

**You control your main loop**. You call pygame functions, they don't call your functions. This gives you greater control when using other libraries, and for different types of programs.

**Does not require a GUI to use all functions**. You can use pygame from a command line if you want to use it just to process images, get joystick input, or play sounds.

**Fast response to reported bugs**. Some bugs are patched within an hour of being reported. Do a search on our mailing list for BUG... you'll see for yourself. Sometimes we suck at bug fixes, but mostly we're pretty good bug fixers. Bug reports are quite rare these days, since a lot of them have been fixed already.

**Small amount of code**. It does not have hundreds of thousands of lines of code for things you won't use anyway. The core is kept simple, and extra things like GUI libraries, and effects are developed separately outside of pygame.

**Modular**. You can use pieces of pygame separately. Want to use a different sound library? That's fine. Many of the core modules can be initialized and used separately.



**Why Quarantine snake**

A **quarantine** is a restriction on the movement of people and goods which is intended to prevent the spread of disease or pests. It is often used in connection to disease and illness, preventing the movement of those who may have been exposed to a communicable disease, but do not have a confirmed medical diagnosis. It is distinct from medical isolation, in which those confirmed to be infected with a communicable disease are isolated from the healthy population. Quarantine considerations are often one aspect of border control.

The concept of quarantine has been known since biblical times, and is known to have been practised through history in various places. Notable quarantines in modern history include that of the village of Eyam in 1665 during the bubonic plague outbreak in England; East Samoa during the 1918 flu pandemic; the 1972 Yugoslav smallpox outbreak, and extensive quarantines applied throughout the world during the 2019–20 coronavirus pandemic.

Ethical and practical considerations need to be considered when applying quarantine to people. Practice differs from country to country. In some countries, quarantine is just one of many measures governed by legislation relating to the broader concept of biosecurity; for example Australian biosecurity is governed by the single overarching *Biosecurity Act 2015*.

The word *quarantine* comes from *quarantena*, meaning "forty days", used in the 14th–15th-centuries Venetian language and designating the period that all ships were required to be isolated before passengers and crew could go ashore during the Black Death plague epidemic; it followed the *trentino*, or thirty-day isolation period, first imposed in 1377 in the Republic of Ragusa, Dalmatia (modern Dubrovnik in Croatia).

Merriam-Webster gives various meanings to the noun form, including "a period of 40 days", several relating to ships, "a state of enforced isolation", and as "a restriction on the movement of people and goods which is intended to prevent the spread of disease or pests". The word is also used as a verb.

Quarantine is distinct from medical isolation, in which those confirmed to be infected with a communicable disease are isolated from the healthy population

Quarantine may be used interchangeably with *cordon sanitaire*, and although the terms are related, *cordon sanitaire* refers to the restriction of movement of people into or out of a defined geographic area, such as a community, in order to prevent an infection from spreading

"If the shiny spot on the skin is white but does not appear to be more than skin deep and the hair in it has not turned white, the priest is to isolate the affected person for seven days. On the seventh day the priest is to examine him, and if he sees that the sore is unchanged and has not spread in the skin, he is to isolate him for another seven daysThe Islamic prophet Muhammad advised quarantine: "Those with contagious diseases should be kept away from those who are healthy." Ibn Sina also recommended quarantine for patients with infectious diseases, especially tuberculosis.

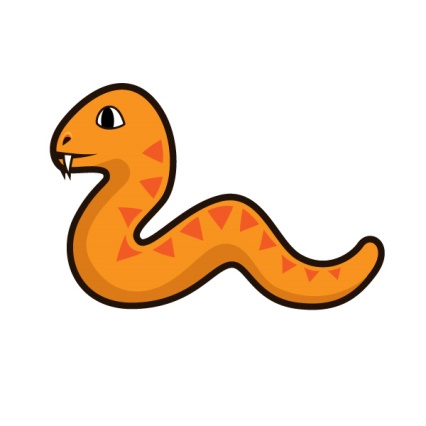
The mandatory hospital quarantine of special groups of patients, including those with leprosy, started early in Islamic history.Between 706 and 707 the sixth Umayyad caliph Al-Walid I built the first hospital in Damascus and issued an order to isolate those infected with leprosy from other patients in the hospital. The practice of mandatory quarantine of leprosy in general hospitals continued until the year 1431, when the Ottomans built a leprosy hospital in Edirne. Incidents of quarantine occurred throughout the Muslim world, with evidence of voluntary community quarantine in some of these reported incidents. The first documented involuntary community quarantine was established by the Ottoman quarantine reform in 1838

The word "quarantine" originates from *quaranten*, the Venetian language form, meaning "forty days". This is due to the 40-day isolation of ships and people practised as a measure of disease prevention related to the plague . Between 1348 and 1359, the Black Death wiped out an estimated 30% of Europe's population, and a significant percentage of Asia's population. Such a disaster led governments to establish measures of containment to handle recurrent epidemics.A document from 1377 states that before entering the city-state of Ragusa in Dalmatia (modern Dubrovnik in Croatia), newcomers had to spend 30 days (a *trentine*) in a restricted place (originally nearby islands) waiting to see whether the symptoms of Black Death would develop. In 1448 the Venetian Senate prolonged the waiting period to 40 days, thus giving birth to the term "quarantine". The forty-day quarantine proved to be an effective formula for handling outbreaks of the plague. Dubrovnik was the first city in Europe to set up quarantine sites such as the Lazzarettos of Dubrovnik where arriving ship personnel were held for up to 40 days. According to current estimates, the bubonic plague had a 37-day period from infection to death; therefore, the European quarantines would have been highly successful in determining the health of crews from potential trading and supply ships.

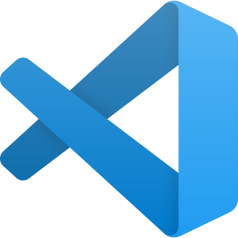
Other diseases lent themselves to the practice of quarantine before and after the devastation of the plague. Those afflicted with leprosy were historically isolated long-term from society, and attempts were made to check the spread of syphilis in northern Europe after 1492, the advent of yellow fever in Spain at the beginning of the 19th century, and the arrival of Asiatic cholera in 1831.

Venice took the lead in measures to check the spread of plague, having appointed three guardians of public health in the first years of the Black Death (1348).The next record of preventive measures comes from Reggio/Modena in 1374. Venice founded the first lazaret (on a small island adjoining the city) in 1403. In 1467 Genoa followed the example of Venice, and in 1476 the old leper hospital of Marseille was converted into a plague hospital. The great lazaret of Marseille, perhaps the most complete of its kind, was founded in 1526 on the island of Pomègues. The practice at all the Mediterranean lazarets did not differ from the English procedure in the Levantine and North African trade. On the arrival of cholera in 1831 some new lazarets were set up at western ports, notably a very extensive establishment near Bordeaux, afterwards turned to another use.

quarantined for 9–15 days. Upon appearance of the plague, the quarantine stations would be militarised and the Ottoman army would be involved in border control Epidemics of yellow fever ravaged urban communities in North America throughout the late-eighteenth and early-nineteenth centuries, the best-known examples being the 1793 Philadelphia yellow fever epidemic and outbreaks in Georgia (1856) and Florida (1888).[[21]](https://en.wikipedia.org/wiki/Quarantine#cite_note-21) Cholera and smallpox epidemics continued throughout the nineteenth century, and plague epidemics affected Honolulu[[22]](https://en.wikipedia.org/wiki/Quarantine#cite_note-22) and San Francisco from 1899 until 1901.[[23]](https://en.wikipedia.org/wiki/Quarantine#cite_note-23) State governments generally relied on the *cordon sanitaire* as a geographic quarantine measure to control the movement of people into and out of affected communities. During the 1918 influenza pandemic, some communities instituted protective sequestration (sometimes referred to as "reverse quarantine") to keep the infected from introducing influenza into healthy populations.



**TEXT EDITOR USED**

**Visual Studio Code**

It is a source-code editor developed by Microsoft for Windows, Linux and macOS. It includes embedded Git and support for debugging, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is highly customizable, allowing users to change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality. The source code is free and open-source, released under the permissive MIT License. The compiled binaries are freeware for any use.

In the Stack Overflow 2019 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool, with 50.7% of 87,317 respondents claiming to use it

Visual Studio Code was announced on April 29, 2015, by Microsoft at the 2015 Build conference. A Preview build was released shortly thereafter.

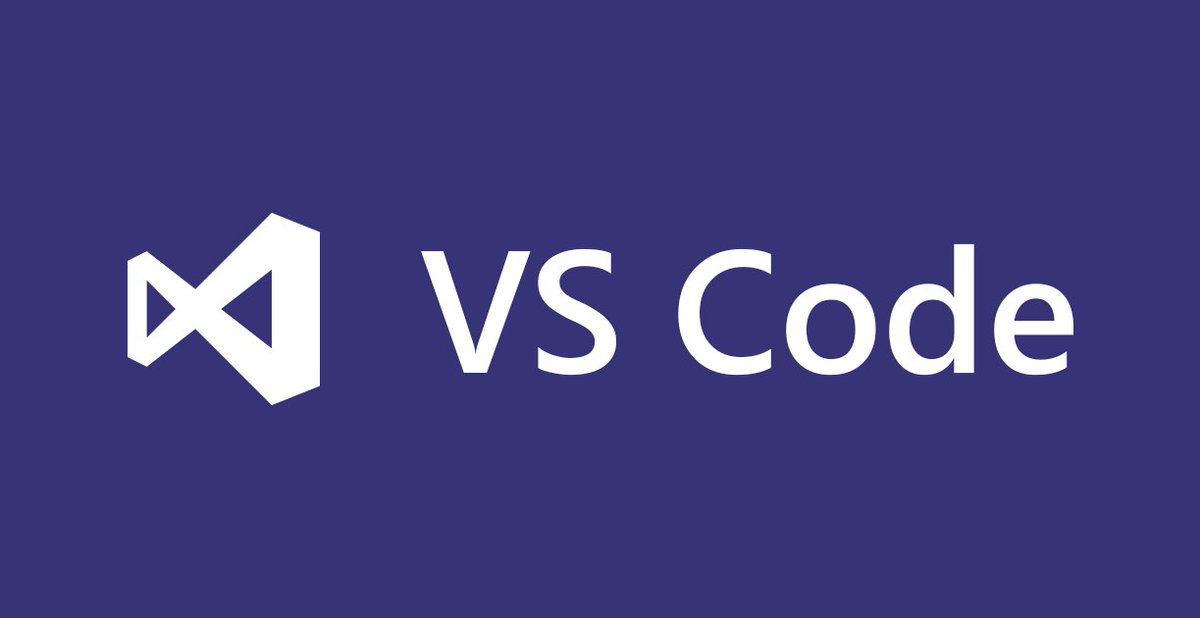
On November 18, 2015, Visual Studio Code was released under the MIT License and its source code posted to GitHub. Extension support was also announced.

On April 14, 2016, Visual Studio Code graduated the public preview stage and was released to web

Visual Studio Code is a source code editor that can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js and C++. It is based on the Electron framework, which is used to develop Node.js web apps that run on the Blink layout engine. Although it uses the Electron framework, the software does not use Atom and instead employs the same editor component (codenamed "Monaco") used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services).

Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse. This allows it to operate as a language-agnostic code editor for any language, contrary to Microsoft Visual Studio which uses the proprietary .sln solution file and project-specific project files. It supports a number of programming languages and a set of features that differs per language. Unwanted files and folders can be excluded from the project tree via the settings. Many of Visual Studio Code features are not exposed through menus or the user interface, but can be accessed via the command palette.

Visual Studio Code can be extended via extensions, available through a central repository. This includes additions to the editor and language support. A notable feature is the ability to create extensions that add support for new languages, themes, and debuggers, perform static code analysis, and add code linters using the Language Server Protocol.



**PROCEDURES**

let’s have a quick look at all the sub-bits that build the Snake Game in Python

1. [Installing Pygame](https://www.edureka.co/blog/snake-game-with-pygame/#install)
2. [Create the Screen](https://www.edureka.co/blog/snake-game-with-pygame/#screen)
3. [Create the Snake](https://www.edureka.co/blog/snake-game-with-pygame/#createthesnake)
4. [Moving the Snake](https://www.edureka.co/blog/snake-game-with-pygame/#move)
5. [Game Over when Snake hits the boundaries](https://www.edureka.co/blog/snake-game-with-pygame/#boundaries)
6. [Adding the Food](https://www.edureka.co/blog/snake-game-with-pygame/#food)
7. [Increasing the Length of the Snake](https://www.edureka.co/blog/snake-game-with-pygame/#increaselength)
8. [Displaying the Score](https://www.edureka.co/blog/snake-game-with-pygame/#score)

## ****Installing Pygame:****

The first thing you will need to do in order to create games using Pygame is to install it on your systems. To do that, you can simply use the following command:

pip install pygame

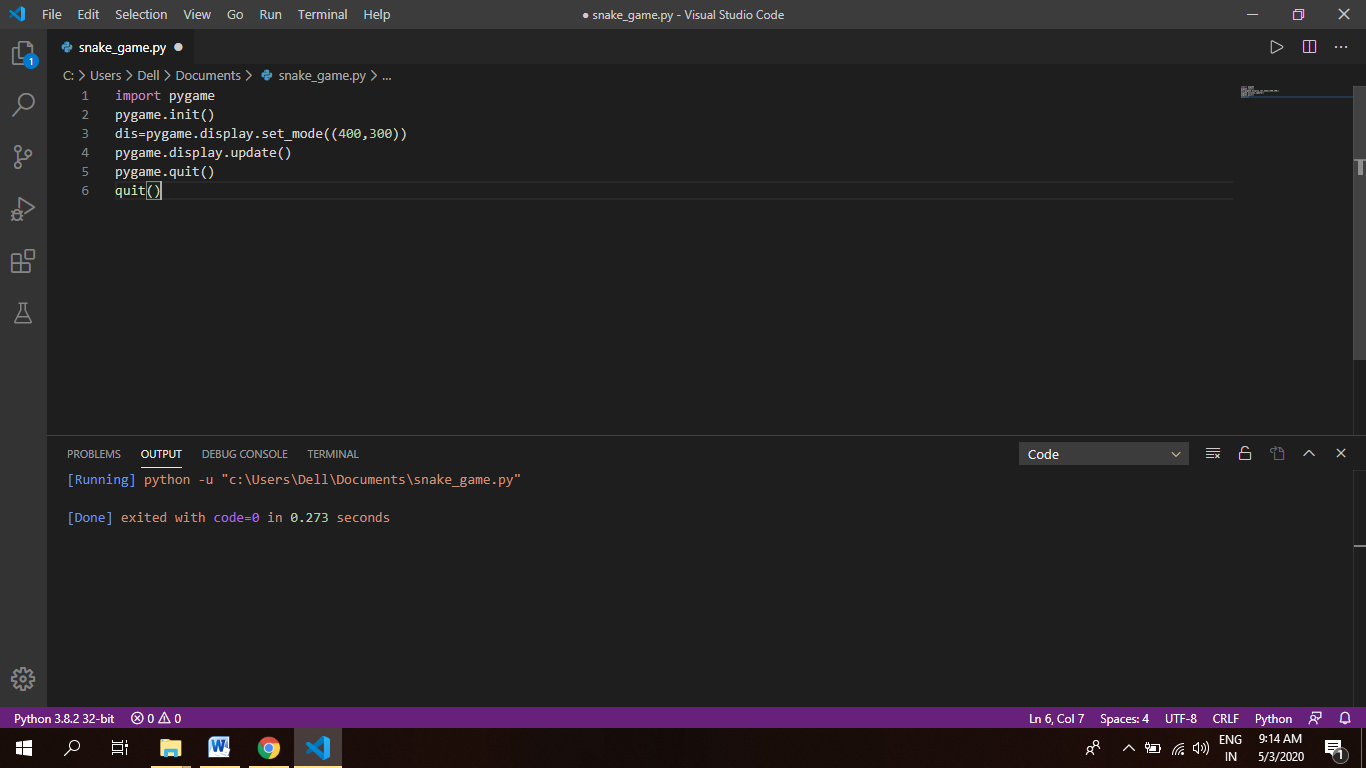
Once that is done, just import Pygame and start off with your game development. Before moving on, take a look at the Pygame functions that have been used in this Snake Game along with their descriptions.

|  |  |
| --- | --- |
| Function | Description |
| init() | Initializes all of the imported Pygame modules (returns a tuple indicating success and failure of initializations) |
| display.set\_mode() | Takes a tuple or a list as its parameter to create a surface (tuple preferred) |
| update() | Updates the screen |
| quit() | Used to uninitialize everything |
| set\_caption() | Will set the caption text on the top of the display screen |
| event.get() | Returns list of all events |
| Surface.fill() | Will fill the surface with a solid color |
| time.Clock() | Helps track time time |
| font.SysFont() | Will create a Pygame font from the System font resources |

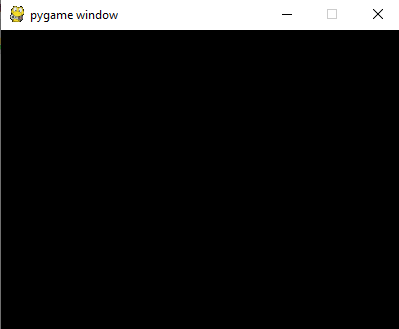
## ****Create the Screen:****

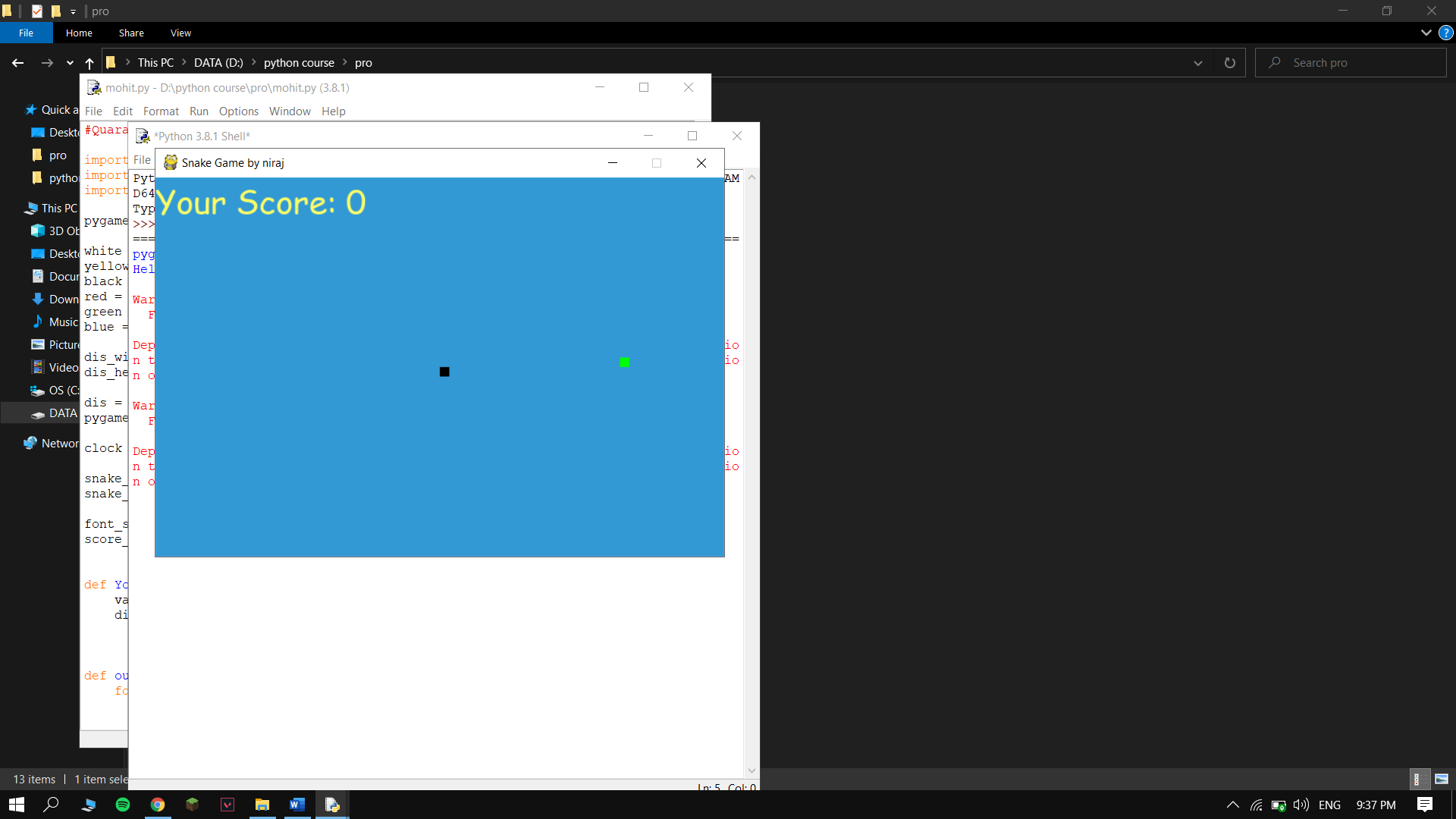
To create the screen using Pygame, you will need to make use of the display.set\_mode() [function](https://www.edureka.co/blog/python-functions). Also, you will have to make use of the init()  and the quit() methods to initialize and uninitialize everything at the start and the end of the code. The update() method is used to update any changes made to the screen. There is another method i.e flip() that works similarly to the update() function. The difference is that the update() method updates only the changes that are made (however, if no parameters are passed, updates the complete screen) but the flip() method redoes the complete screen again.

**CODE:**



**OUTPUT:**





**CONCLUSION**

This Project gave me the best project experience of my life till now as I have never done anything like this before and for this I would like my teacher Mr. Arun Mittal for assigning me this wonderful project with a wonderful topic that is problem management. Through this project I came to know about the working of python programming language and pygames.

Thank you for your anticipation.

**BIBLIOGRAPHY/REFERENCES**

In this project I took references from:-

1. SUMITA ARORA BOOKS
2. GOOGLE
3. YOUTUBE
4. GUIDELINES FROM Mr. Arun Mittal

**5.**

[https://www.w3schools.in](https://www.w3schools.in  )





