

# **SPACE INVADERS**

Submitted in partial fulfillment of the requirements for the award of  
Bachelor of Engineering Degree in Computer Science and Engineering

By

**PENMETSA YASASWI SATYA PAVAN VARMA**

**(Reg. No. 37110562)**

**NIMMAKAYALAA PARDHA VENKATA SAI**

**(Reg. No. 37110511)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SCHOOL OF COMPUTING**

**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,**

**CHENNAI – 600119, TAMILNADU**

**MARCH 2021**



# SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

## SCHOOL OF COMPUTING

### BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of **PENMETSA YASASWI SATYA PAVAN VARMA**(Reg.No.37110562) and **NIMMAKAYALAA PARDHA VENKATA SAI**(Reg.No.37110511) who carried out the project entitled as “**SPACE INVADERS**” under our supervision from JULY 2020 to AUGUST 2021.

#### Internal Guide

DR.G.MATHIVANAN ,M.TECH.,PHD.,

#### Head of the Department

Dr. S. VIGNESHWARI, M.E., PHD.,

Submitted for Viva voice Examination held on\_\_\_\_\_

Internal Examiner

External Examiner

## DECLARATION

We, **PENMETSA YASASWI SATYA PAVAN VARMA**(Reg.No.37110562) and **NIMMAKAYALAA PARDHA VENKATA SAI**(Reg.No.37110511) here by declare that the Professional Training Report on “**SPACE INVADERS**” done by us under the guidance of **DR.G.MATHIVANAN,M.TECH.,PHD.**, at Sathyabama institute of science and technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

**DATE:**

**PLACE:**

**CANDIDATE**

**SIGNATURE OF THE**

## ACKNOWLEDGEMENT

- We are pleased to acknowledge our sincere thanks to Board of Management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. we are grateful to them.
- We convey our thanks to **Dr.T.SASIKALA PhD.**, Dean, School of Computing and **Dr.S. VIGNESHWARI, M.E., PhD.**, Head of the Department, Department of Computer Science and Engineering for providing necessary support and details at the right time during the progressive reviews.
- We would like to express our sincere and deep sense of gratitude to our Project Guide **DR.G.MATHIVANAN,M.TECH.,PHD** for his amazing and valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.
- We wish to express our thanks to all Teaching and Non-teaching staff members of the Department of **COMPUTER SCIENCE AND ENGINEERING** who were helpful in many ways for the completion of the project.





## CERTIFICATE OF TRAINING

JAVA PROGRAMMING

**NIMMAKAYALA PARDHA VENKATA SAI** from **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** has successfully undergone a two weeks training and Internship on Python. The training program lasted for two weeks from 16<sup>th</sup> December, 2019 to 26<sup>th</sup> December, 2019.

We wish NIMMAKAYALA PARDHA VENKATA SAI all the best for future endeavours

A handwritten signature in black ink, appearing to read "Sarvesh".

**Sarvesh Agrawal**  
**Founder & CEO**

Date of certification: 26-12-2019

Certificate Number: 541DB74B-AEB4-7D25-593A-B8816054D20Y

For certificate authentication please visit [https://trainings.internshala.com/verify\\_certificate](https://trainings.internshala.com/verify_certificate)

## **ABSTRACT**

Computer gaming is a key component of the rapidly growing entertainment industry. While building computer games has typically been a commercial endeavor, we believe that designing and constructing a computer game is also a useful activity for educating students about geometric modeling and computer graphics. In particular, students are exposed to the practical issues surrounding topics such as geometric modeling, rendering, collision detection, character animation and graphical design. Moreover, building an advanced game provides students exposure to the real-world side of software engineering that they are typically shielded from in the standard computer class. In this paper, we describe our experiences with teaching a computer science class that focuses on designing and building the best game possible in the course of a semester. The paper breaks down a typical game into various components that are suited for individual student projects and discusses the use of modern graphical design tools such as Maya in building art for the game. We conclude with a rough timeline for developing a game during the course of a semester and review some of the lessons learned during the three years we have taught the class.

# TABLE OF CONTENTS

## TITLE

|                                               |            |
|-----------------------------------------------|------------|
| <b>ABSTRACT</b>                               | <b>i i</b> |
| <b>LIST OF FIGURES</b>                        | <b>v</b>   |
| <b>LIST OF TABLES</b>                         | <b>v</b>   |
| <b>1. INTRODUCTION</b>                        |            |
| 1.1 PYTHON PROGRAMMING                        | 1          |
| 1.1.1 OVER VIEW OF PROJECT                    | 1          |
| 1. 1. 2 FEATURES                              | 1          |
| 1.1.3 APPLICATIONS                            | 2          |
| 1.1.4 LITERATURE REVIEW                       | 4          |
| 1.1.5 ENVIRONMENT SETUP                       | 4          |
| 1.1.6 DISADVANTAGES                           | 6          |
| <br>                                          |            |
| <b>2. ALGORITHMS AND DESIGNS</b>              |            |
| 2.1 GENERAL                                   | 7          |
| 2.2 OVERVIEW                                  | 7          |
| 2.2.1 RUNNING PYTHON                          | 8          |
| <b>2.3</b> INTEGRATED DEVELOPMENT ENVIRONMENT | 9          |
| 2.4 PYTHON IDENTIFIERS                        | 9          |
| 2.5 RESERVED WORDS                            | 10         |
| 2.6 LINES AND INDENTATION                     | 10         |
| 2.7 MULTI-LINE STATEMENT                      | 11         |
| 2.8 QUOTATION IN PYTHON                       | 11         |
| 2.9 COMMENTS INPYTHON                         | 11         |

|          |                                   |     |
|----------|-----------------------------------|-----|
| <b>3</b> | <b>SYSTEM IMPLEMENTATION</b>      |     |
| 3.1      | REQUIREMENTS                      | 13  |
| 3.2      | SOFT WARE PLATFORM                | 13  |
| 3.3      | GUI PROGRAMMING                   | 14  |
| 3.4      | GETTING STARTED WITH TKINTER      | 14  |
| 3.4.1    | TKINTER WIDGETS                   | 16  |
| 3.4.2    | STANDARD ATTRIBUTES               | 18  |
| 3.4.3    | GEOMETRY MANAGEMENT               | 18  |
| 3.5      | SNAPSHOTS AND EXPLANATIONS        | 19  |
| <br>     |                                   |     |
| <b>4</b> | <b>CONCLUSION AND FUTURE WORK</b> |     |
| 4.1      | SUMMARY                           | 22  |
| 4.2      | CONCLUSION                        | 22  |
| 4.3      | FUTURE WORKS                      | 2 2 |
|          | <b>APPENDIX</b>                   |     |
|          | SOURCE CODE                       | 23  |
|          | OUTPUT                            | 37  |
|          | <b>REFERENCES</b>                 | 38  |

## LIST OF FIGURES

| Fig. No | TITLE                                | PAGE No. |
|---------|--------------------------------------|----------|
| 3.1     | TKINTER WINDOW                       | 1 6      |
| 3.2     | CREATING THE ITEMS                   | 1 9      |
| 3.3     | CODE FOR CREATING WIDGETS            | 1 9      |
| 3.4     | CODE FOR TRAVELLING TICKETING SYSTEM | 2 0      |
| 3.5     | OUTPUT FOR WIDGETS                   | 2 0      |
| 3.6     | TOTAL OUTPUT SCREEN FOR WIDGETS      | 2 1      |

## LIST OF TABLES

| TABLE NO. | TITLE                        | PAGE NO |
|-----------|------------------------------|---------|
| 2.1       | PYTHON ENVIRONMENT VARIABLES | 8       |
| 2.2       | PYTHON COMMAND LINE OPTIONS  | 9       |
| 2.3       | PYTHON KEYWORDS              | 11      |

# CHAPTER 1

## INTRODUCTION

### 1.1 PYTHON PROGRAMMING

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas the other languages use punctuations. It has fewer syntactical constructions than other languages. Python is Interpreted- Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is as same as PERL and PHP. Python is interactive- You can set at a Python prompt and interact with the interpreter directly to write your programs. Python is Object-Oriented- Python supports Object-Oriented style or technique of programming that encapsulates code within objects. Python is a Beginner's Language- Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

#### 1.1.1 OVERVIEW OF PROJECT

- This project is about creating train ticket management systems in python programming. In this thesis, a front end page is created using Graphic User Interface in Python Programming and the back end which is used to store the data is done by Sqlite3. The front end page shows the details of the train ticket such as the source and destination, type of the ticket an price of ticket.

#### 1.1.2 FEATURES

Python's features include-

**Easy to learn** – Python has few keywords, simple structure, and a clearly defined syntax. This allows a student to pick up language quickly.

**Easy to read** – Python code is more clearly defined and visible to the eyes.

**Easy to maintain** – Python's source code is easy to maintain.

**A broad standard library** – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

**Interactive Mode** – Python has support for an interactive mode which allows interactive mode which allows interactive testing and debugging of snippets of code.

**Portable** – Python can run on a wide variety of hardware platforms and has the same interface on the platforms.

**Extendable** – You can add low level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

**Databases** – Python provides interfaces to all major commercial databases.

**GUI Programming** – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as windows MFC, Macintosh, and the X window system of Unix.

**Scalable** – Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features a few are listed below –

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

### **1.1.3 APPLICATIONS**

Web and Internet Development:

Python offers many choices for web development:

- Frameworks such as Django and Pyramid.
- Micro-frameworks such as Flask and Bottle.
- Advanced content management systems such as Plone and django CMS.

Python's standard library supports many Internet protocols:

- HTML and XML
- JSON
- E-mail processing.

- Support for FTP, IMAP, and other Internet protocols.
- Easy-to-use socket interface.

### **Scientific and Numeric:**

Python is widely used in scientific and numeric computing:

- SciPy is a collection of packages for mathematics, science, and engineering
- Pandas is a data analysis and modeling library.
- The Software Carpentry Course teaches basic skills for scientific computing, running boot camps and providing open-access teaching materials.

### **Education:**

Python is a superb language for teaching programming, both at the introductory level and in more advanced courses.

### **Desktop GUIs:**

The Tk GUI library is included with most binary distributions of Python.

Some toolkits that are usable on several platforms are available separately:

- wxWidgets
- Kivy, for writing multitouch applications.
- Qt via pyqt or pyside

### **Platform-specific toolkits are also available:**

- GTK+
- Microsoft Foundation Classes through the win32 extensions.

### **Software development:**

Python is often used as a support language for software developers, for build and management, testing, and in many other ways.

- SCons for build control.
- Buildbot and Apache Gump for automated continuous compilation and testing.
- Roundup or Trac for bug tracking and project management.

### **Business Applications**

Python is also used to build ERP and e-commerce systems:

- Odoo is an all-in-one management software that offers a range of business applications that form a complete suite of enterprise management applications.
- Tryton is a three-tier high-level general-purpose application platform.

#### **1.1.4 Literature Review**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, and Unix shell and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress. Python 1.0 was released in November 1994. Python 2.0 was released in 2008. Meanwhile, Python 3.0 was released in 2008. Python 3 is not backward compatible with Python 2. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules. Python 3.5.1 is the latest version of Python 3.

#### **1.1.5 Environment Setup**

- Python 3 is available for Windows, Mac OS and most of the flavors of Linux operating system. Even though Python 2 is available for many other OSs, Python3 support either has not been made available for them or has been dropped.
- Windows platform-Binaries of latest version of Python3 are available on the download page.

The following different installation options are available.

- Windows x86-64 embeddable zip file
- Windows x86-64 executable installer
- Windows x86-64 web-based installer
- Windows x86 embeddable zip file
- Windows x86 executable installer

- Windows x86 web-based installer

**Linux platform-** Different flavors of Linux use different package managers for installation of new packages.

389039On Ubuntu Linux, Python3 is installed using the following command from the terminal:

```
$sudo apt-get install python3-minimal
```

Mac OS- Download Mac OS installers from this

URL – <https://www.python.org/downloads/mac-osx/>

- Mac OSX 64-bit/32-bit installer -python-3.5.1-macosx10.6.pkg
- Mac OSX 32-bit i386/PPC installer-python-3.5.1-macosx10.5.pkg

Double click this package file and follow the wizard instructions to install.

The most up to date and current source code, binaries, documentation, news, etc., is available on the official website of Python.

Official website of Python – <https://www.python.org/>

*Setting up path-* programs and other executable files can be in many directories. Hence, the operating systems provide a search path that lists the directories that it searches for executables.

The important features are-

- The path is stored in an environment variable, which is a named string maintained by the operating system. This variable contains information available to the command shell and other programs.
- The path variable is named as PATH in Unix or Path in Windows
- In Mac OS, the installer handles the path details. To invoke the Python interpreter from any particular directory, you must add the Python directory, you must add the Python directory to your path.

Setting Path at Unix/Linux-To add the Python directory to the path for a particular session in Unix-

In the csh shell – type setenv PATH "\$PATH:/usr/local/bin/ppython3" and press Enter.

In the bash shell(Linux) – type export PYTHONPATH=/usr/local/bin/python3.4 and press Enter.

In the sh or ksh shell – type PATH="\$PATH:/usr/local/bin/python3" and press Enter.

Setting Path at Windows-To add the Python directory to the path for a particular session in Windows-

At the command prompt – type path %path%C:\Python and press Enter.

#### 1.1.6 **DISADVANTAGES**

Python has varied advantageous features, and programmers prefer this language to other programming languages because it is easy to learn and code too. However, this language has still not made its place in some computing arenas that includes Enterprise Development Shops. Therefore, this language may not solve some of the enterprise solutions, and limitations include-

Difficulty in Using Other Languages: The Python lovers become so accustomed to its features and its extensive libraries, so they face problem in learning or working on other programming languages. Python experts may see the declaring of cat “values” or variable “types”, syntactic requirements of adding curly braces or semi colons as an onerous task.

- Weak in Mobile Computing: Python has made its presence on many desktop and server platforms, but it is seen as a weak language for mobile computing. This is the reason very few mobile applications are built in it Carbonnelle.
- Gets Slow in Speed: Python executes with the help of an interpreter instead of the compiler, which causes it to slow down because compilation and execution help it to work normally. On the other hand, it can be seen that it is fast foe many web applications too.
- Run-time Errors: The Python language is dynamically typed so it has many design restrictions that are reported by some Python developers. It is even seen that it requires more testing time, and the errors how up when the applications are finally run.
- Underdeveloped Database Access Layers: As compared to the popular technologies like JDBC and ODBC, the Python’s database access layer is found to be bit underdeveloped and primitive. However, it cannot be applied in the enterprises that need smooth interaction of complex legacy data.

## **CHAPTER 2**

### **ALGORITHMS AND DESIGNS**

#### **2.1 GENERAL**

To know the Python Programming and to run the programs in Python we need to know about Python completely. We have different types of topics in Python to study, namely, Operators, Expressions, Control Statements, Data structures, Functions, Packages, Object Oriented Programming, Inheritance and Polymorphism, Exception Handling, Standard Libraries, Testing.

## 2.2 OVERVIEW

Here are important environment variables, which are recognized by Python-

**Table 2.1 Python Environment Variables**

| S. no: | Variable & Description                                                                                                                                                                                                                                                                                                     |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1.     | <b>PYTHONPATH</b><br>It has a role similar to PATH. This variable tells the Python interpreter where to locate the module files imported into a program. It should include the Python source library directory and the directories containing Python source code. PYTHONPATH is sometimes present by the Python installer. |
| 2.     | <b>PYTHONSTARTUP</b><br>It contains the path of an initialization file containing Python source code. It is executed every time you start the interpreter. It is named as .pythonrc.py in Unix and it contains commands that load utilities or modify PYTHONPATH                                                           |
| 3.     | <b>PYTHONCASEOK</b><br>It is used in Windows to instruct Python to find the first case-insensitive match in an import statement. Set this variable to any value to activate it.                                                                                                                                            |
| 4.     | <b>PYTHONHOME</b><br>It is an alternative module search path. It is usually embedded in the PYTHONSTARTUP or PYTHONPATH directories to make switching module libraries easy.                                                                                                                                               |

### 2.2.1 RUNNING PYTHON

There are three different ways to start Python –  
Interactive Interpreter:

You can start Python from Unix, DOS, or any other system that provides you a command-line interpreter or shell window.

Enter *python* the command line.

Start coding right away in the interactive interpreter.

\$python                      #Unix/Linux

Or

Python%                      #Unix/Linux

Or

C:>python                      #Windows/DOS

Here is the list of all the available command line options –

**Table 2.2 Python Command Line Options:**

| S. No. | Option & Description                                                                                |
|--------|-----------------------------------------------------------------------------------------------------|
| 1.     | -d<br>Provide debug output                                                                          |
| 2.     | -O<br>Generate optimized bytecode(resulting in .pyo files)                                          |
| 3.     | -S<br>Do not run import site to look for Python paths on startup                                    |
| 4.     | -v<br>Verbose output (detailed trace on import statements)                                          |
| 5.     | -X<br>Disable class-based built-in exceptions(just use strings); obsolete starting with version 1.6 |
| 6.     | -c cmd<br>Run Python script sent in as cmd string                                                   |
| 7.     | File<br>run Python script from given file                                                           |

Script from the Command-line:

A Python script can be executed at the command line by invoking the interpreter on your application, as shown in the following example.

\$python script.py                      #Unix/Linux

or

```
python% script.py          #Unix/Linux
or
C:>python script.py        #Windows/DOS
```

## 2.3 INTEGRATED DEVELOPMENT ENVIRONMENT

You can run Python from Graphical User Interface(GUI) environment as well, if you have a GUI application on your system that supports Python.

- Unix – Idle is the very first Unix IDE for Python.
- Windows – PythonWin is the first Windows interface for Python and is an IDE with GUI.
- Macintosh – The Macintosh version of Python along with the IDLE IDE is available from the main website, downloadable as either MacBinary or BinHex'd files.
- If you are not able to set up the environment properly, then you can take the help of your system admin. Make sure the Python environment is properly set up and working perfectly fine.

## 2.4 PYTHON IDENTIFIERS

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore ( `_` ) followed by zero or more letters, underscores and digits(0 to 9)

Python does not allow punctuation characters such as `@`, `$`, and `%` within identifiers. Python is a case sensitive programming language. Thus, `Manpower` and `manpower` are two different identifiers in Python.

Here are naming conventions for Python identifiers-

- Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

## 2.5 RESERVED WORDS:

### Table 2.3:Python Keywords

The following list shows the Python keywords. These are reserved words and you cannot use them as constants or variables or any other identifier names. All the Python keywords contain lowercase letters only.

|          |        |        |
|----------|--------|--------|
| Class    | Global | Raise  |
| Continue | If     | Return |
| Def      | Import | Try    |
| Del      | In     | While  |
| Elif     | Is     | With   |
| Else     | Lambda | Yield  |
| Except   |        |        |

## 2.6 LINES AND INDENTATION

Python does not use braces ({} ) to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True:
    print("True")
```

```
else:
    print("False")
```

However, the following block generates an error –

```
if True:
    print("Answer")
    print("True")
```

```
else:
    print("Answer")
    print("False")
```

Thus, in Python all the continuous lines indented with the same number of spaces would form a block.

## 2.7 MULTI-LINE STATEMENTS

Statements in Python typically end with a new line. Python, however, allows the use of the line continuation character (\) to denote that the line should continue. For example –

```
total = item_one+\
```

```
item_two+\nitem_three
```

The statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example –

```
days=['Monday','Tuesday','Wednesday','Thursday','Friday']
```

## 2.8 QUOTATION IN PYTHON

Python accepts single ('), double (") and trip (''' or ''') quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal –

```
word='word'
```

```
sentence=" This is a sentence."
```

```
paragraph=""" This is a paragraph. It is made up of multiple lines and sentences."""
```

## 2.9 COMMENTS IN PYTHON

A hash sign (#) that is not inside a string literal is the beginning of a comment. All characters after the #, up to the end of the physical line, are part of the comment and the Python interpreter ignores them.

```
# !/usr/bin/python3
```

```
# First comment
```

```
print(" Hello, Python!") # Second comment
```

This produces the following result –

```
Hello, Python!
```

### *Using Blank Lines*

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with **-h** –

```
$python -h
```

```
usage: python [option]...[-c cmd ] -m mod [file|-] [arg]...
```

Options and arguments ( and corresponding environment variables):

- c cmd : program passed in as string(terminates option list)
- d :debug output from parser (also PYTHONDEBUG=x)
- E :ignore environment variables (such as PYTHONPATH)
- h :print this help message and exit

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic

## **CHAPTER 3**

### **SYSTEM IMPLEMENTATION**

#### **3.1 REQUIREMENTS**

Processors:

- Intel core i5 processor 4300M at 2.60 GHz or 2.59 GHz (1 socket, 2 cores, 2 threads per core), 8GB of DRAM
- Intel Xeon processor E5-2698 v3 at 2.30 GHz (2 sockets, 16 cores each, 1 thread per core), 64 GB of DRAM
- Intel Xeon Phi processor 7210 at 1.30 GHz (1 socket, 64 cores, 4 threads per core), 32 GB of DRAM, 16 GB of MCDRAM(flat mode enabled)
- 
- Operating systems : Windows 10, Mac OS, and Linux

### 3.2 SOFTWARE PLATFORM

Installing Python software is done in minutes. All we need to install the software is too search in Google. Type the link <https://www.python.org/> in search box and enter.

Assume we have installed Python on the Windows OS. You can start Python in a command window by typing python at the command prompt or by using IDLE. IDLE(Interactive Development Environment) is an integrated development environment(IDE) for Python. You can create, open, save, edit, and run Python programs in IDLE. Both the command-line Python interpreter and IDLE are available after Python is installed on your machine. Note that Python (command line) and IDLE can also be accessed directly from Windows Start button by searching for Python (command line) or IDLE (Python GUI) on Windows 10 or Vista. After Python starts, you will see the symbol >>>. This is the Python statement prompt and Python statement prompt >>> it is where you can enter a Python statement.*Creating Python Source Code Files*-Entering Python statements at the statement prompt>>> is a convenient, but the statements are not saved. To save statements for later use, you can create a text file to store the statements and use the following command to execute the statements in the file:

*Python filename.py*

The text file can be created using a text editor such as Notepad. The filename , is called a Python source file or script file, or module. By connection, Python files are named with the extension .py.

Running a Python program from script file is known as running Python in script mode. Typing a statement at the statement prompt>>> and executing it is called running Python in interactive mode.

### 3.3 GUI PROGRAMMING

Python provides various options for developing graphical user interfaces (GUIs). The most important features are listed below:

- Tkinter – Tkinter is the Python interface to the Tk GUI toolkit shipped with Python.
- WxPython – This is an open – source Python interface for wxWidgets GUI toolkit.
- PyQt – This is also a Python interface for a popular cross – platform Qt GUI library.
- JPython – Jpython is a Python port for java which gives scripts seamless access which gives Python scripts seamless access to the Java class libraries on the local machine <http://www.jpython.org>

There are many other interfaces available, which you can find them on the net.

### 3.4 GETTING STARTED WITH TKINTER

Tkinter enables you to develop GUI programs and is excellent pedagogical tool for learning object-oriented programming.

There are many GUI modules available for developing GUI programs in Python. You have used the turtle module for drawing geometric shapes. Turtle is easy to use and is an effective pedagogical tool for introducing the fundamentals of programming to beginners. However, you cannot use turtle to create graphical user interfaces. Tkinter is not only a useful tool for developing GUI projects, but it is also a valuable pedagogical tool for learning object-oriented programming.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps –

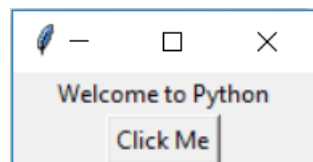
- Import the Tkinter module.
- Create the GUI application main window.
- Add one or more widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

The tkinter module contains the classes for creating GUIs. The Tk class creates a window for holding GUI widgets (i.e., visual components).

Consider the following program to understand Tkinter:

```
from tkinter import *    # import all definitions from tkinter
window=Tk()              #create a window
label=Label(window,text="Welcome to Python")    #Create a label
button=Button(window,text="Click Me")           #Create a button
label.pack()                                           #place the
label in the window
button.pack()                                           #place the
button in the window
window.mainloop()                                     # create an event
loop
```

**When u run the program, a label and a button appear in the Tkinter window, as shown below**



**Fig 3.1 Tinkter Window**

Whenever you create a GUI-based program in Tkinter, you need to import the tkinter module and create a window by using the Tk class. Recall that asterisk (\*) imports all definitions for classes, functions, and constants from the tkinter module to the program. Tk() creates an instance of a window. Label and Button are Python Tkinter widget classes for creating labels and buttons. The first argument of a widget class is always the parent container (i.e., the container in which the widget will be placed). The statement

*label=Label (window, text="Welcome to Python")*

constructs a label with the text Welcome to Python that is contained in the window.

The statement

*label.pack()*

places label in the container using a pack manager. In this example, the pack manager packs the widget in the window row by row.

Tkinter GUI programming is event driven. After the user interface is displayed, the program waits for user interactions such as mouse clicks and key presses. This is specified in the following statement

```
window.mainloop()
```

The statement creates an event loop. The event loop processes events continuously until you close the main window.

### **3.4.1 Tkinter Widgets:**

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table –

**Table 3.1 Types of widgets in Tkinter**

| S.No: | Operator & Description                                                                                                                       |
|-------|----------------------------------------------------------------------------------------------------------------------------------------------|
| 1     | Button:<br>The Button widget is used to display buttons in your application.                                                                 |
| 2     | Canvas:<br>The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.                     |
| 3     | Checkbutton:<br>The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time. |
| 4     | Entry:<br>The Entry widget is used to display a single- line text field for accepting values from a user.                                    |
| 5     | Frame:<br>The Frame widget is used as a container widget to organize other widgets.                                                          |
| 6     | Label:<br>The Label widget is used to provide a single- line caption for other widgets. It can also contain images.                          |
| 7     | Listbox:<br>The Listbox widget is used to provide a list of options to a user.                                                               |
| 8     | Menubutton:                                                                                                                                  |

|    |                                                                                                                                                |
|----|------------------------------------------------------------------------------------------------------------------------------------------------|
|    | The Menubutton widget is used to display menus in your application.                                                                            |
| 9  | Menu:<br>The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.                        |
| 10 | Message:<br>The Message widget is used to display multiline text fields for accepting values from a user.                                      |
| 11 | Radiobutton:<br>The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time. |
| 12 | Scale:<br>The Scale widget is used to provide a slider widget.                                                                                 |
| 13 | Scrollbar:<br>The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.                                 |
| 14 | Text:<br>The Text widget is used to display text in multiple lines.                                                                            |
| 15 | Toplevel:<br>The Toplevel widget is used to provide a separate window container.                                                               |
| 16 | Spinbox:<br>The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.   |
| 17 | PanedWindow:<br>A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.                 |
| 18 | LabelFrame:<br>A label frame is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.  |
| 19 | tkMessageBox:<br>This module is used to display message boxes in your applications.                                                            |

### 3.4.2 STANDARD ATTRIBUTES:

Let us look at how some of their common attributes, such as sizes, colors and fonts are specified.

- Dimensions
- Colors
- Fonts
- Anchors
- Relief styles
- Bitmaps
- Cursors

### **3.4.3 GEOMETRY MANAGEMENT:**

All Tkinter widgets have access to the specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes:

Pack, grid, and place.

- The pack() Method – This geometry manager organizes widgets in blocks before placing them in the parent widget.
- The grid() Method – This geometry manager organizes widgets in a table – like structure in the parent widget.
- The place() Method – This geometry manager organizes widgets by placing them in a specific position in the parent widget.

### **3.5 SNAPSHOTS AND EXPLANTION:**

Firstly, we should install the Python software from the web and open IDLE and create python program and explain how it works with snapahots.

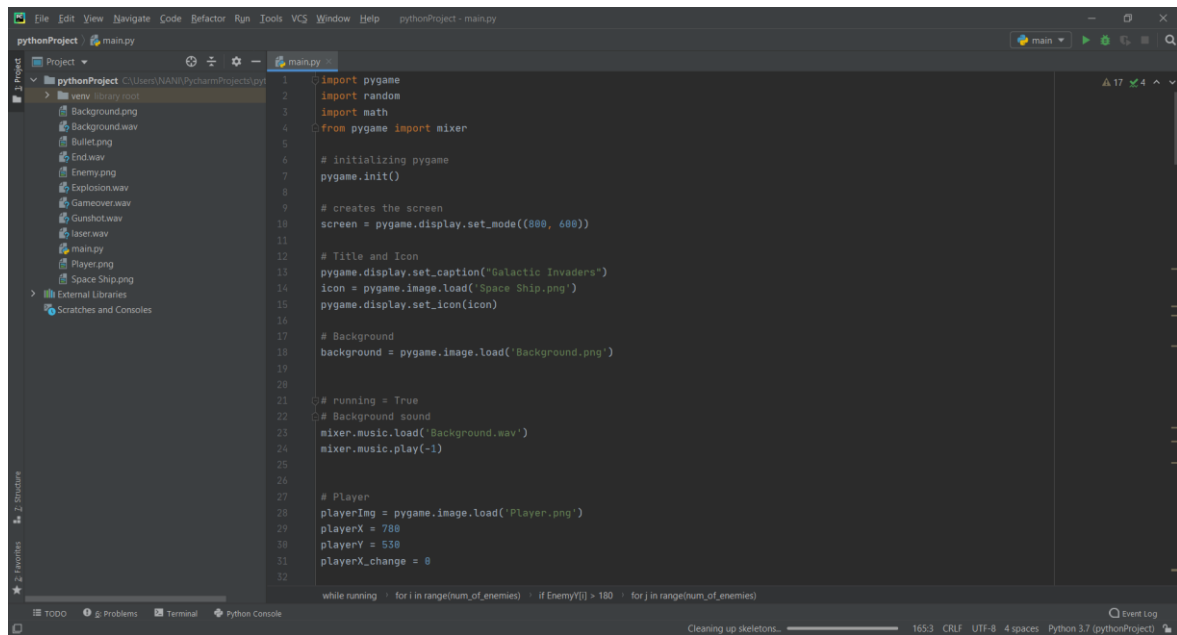


Fig 3.2 INITIALIZING

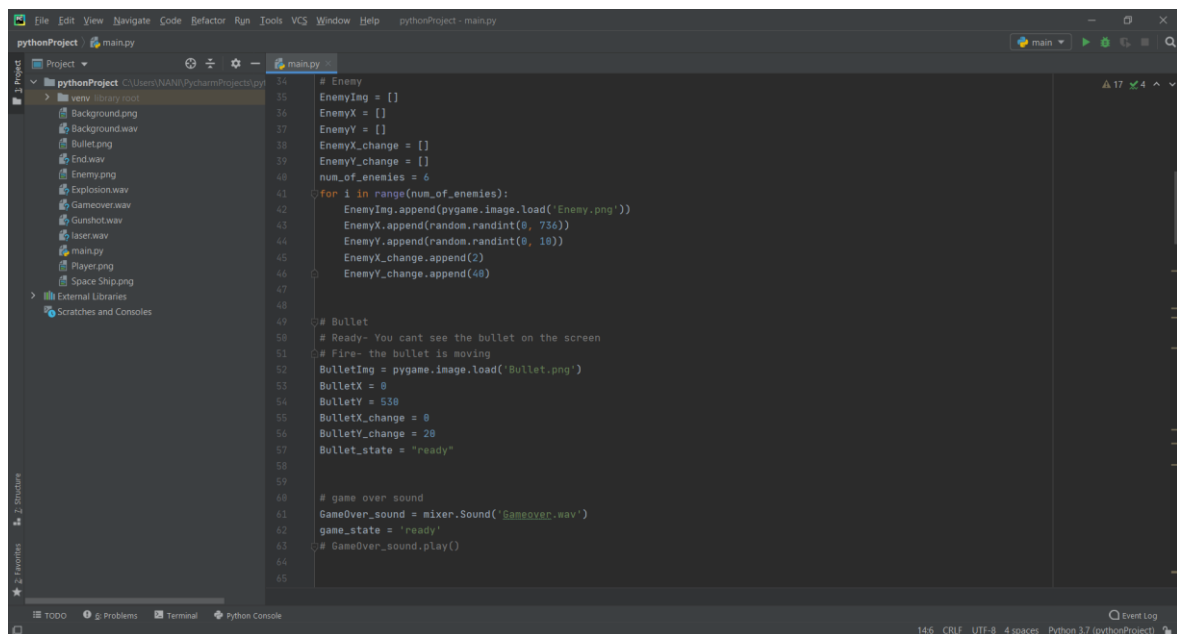
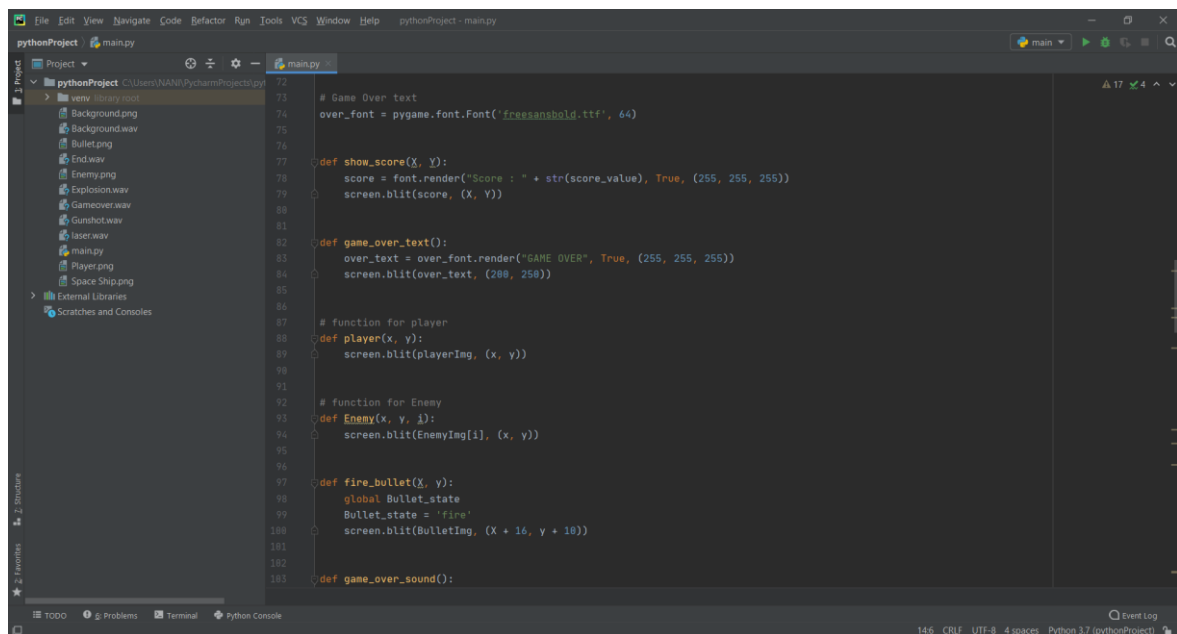
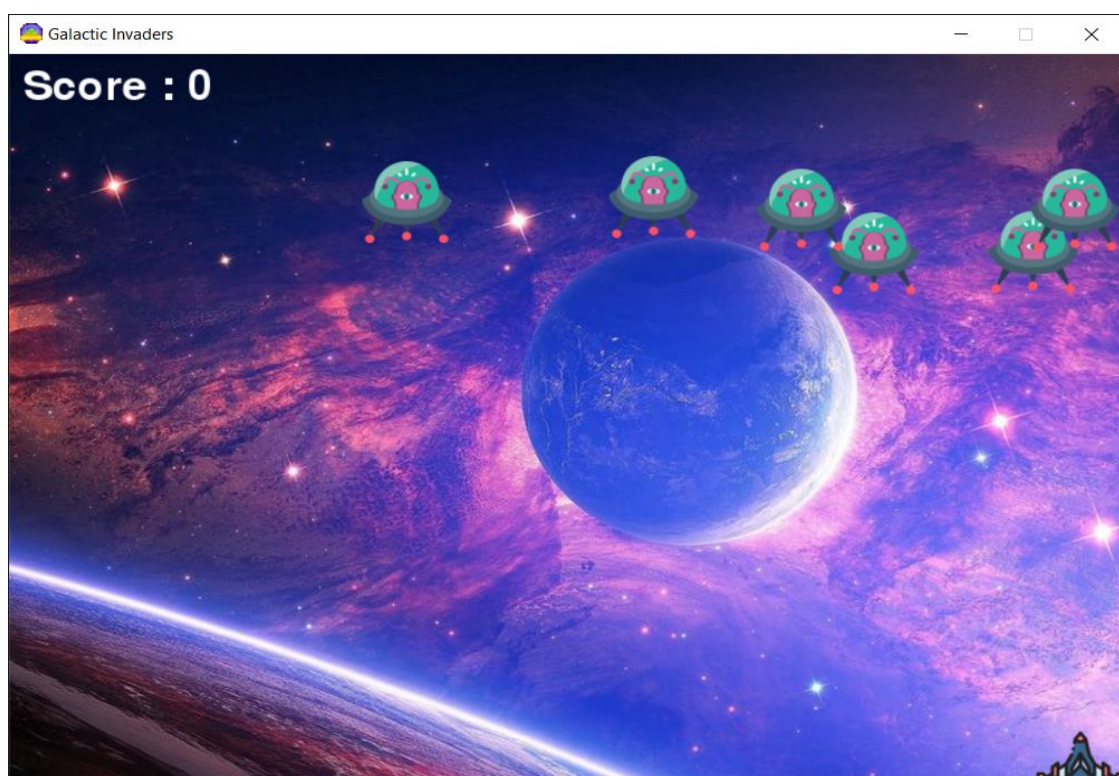


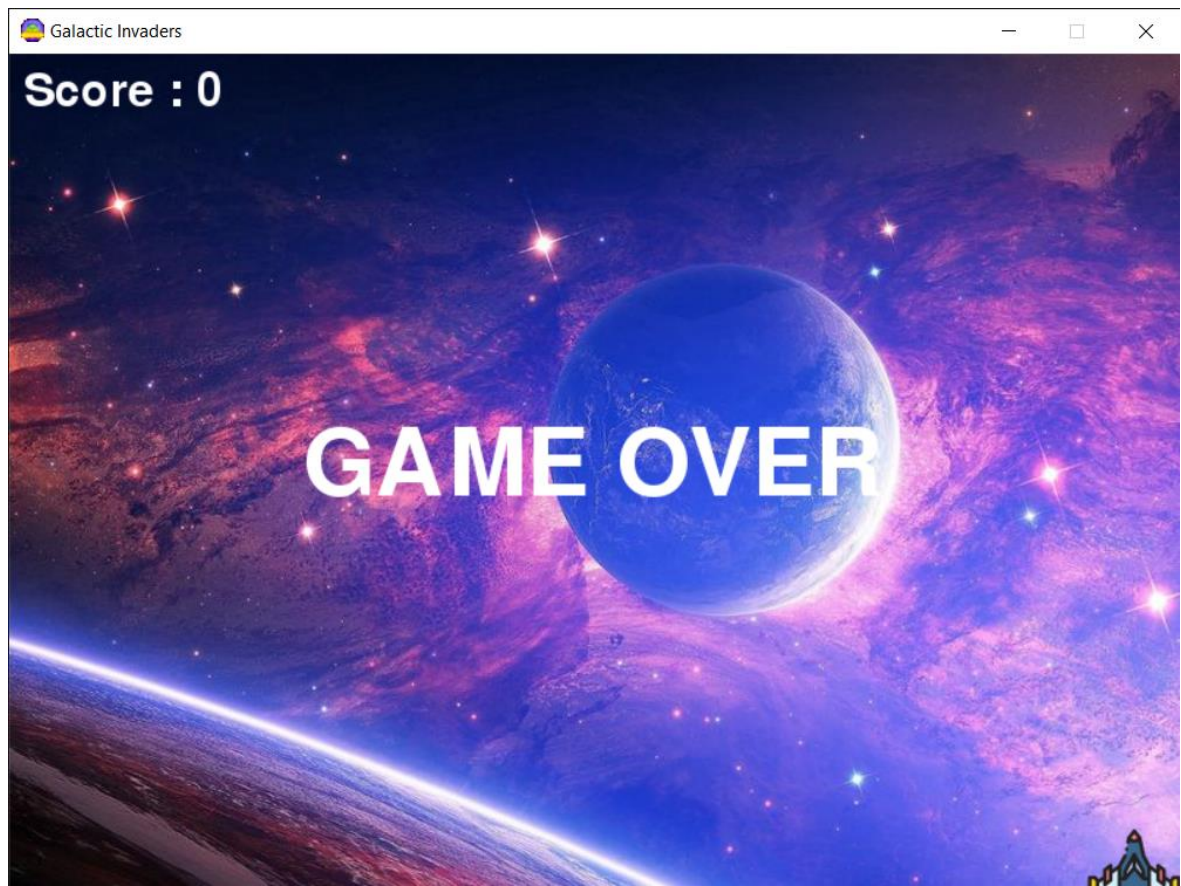
Fig 3.3 CODE FOR CREATING PLAYER AND ENEMY



**Fig 3.4 CODE FOR DEFINING FUNCTIONS**



**Fig 3.5 OUTPUT AS GAME STARTS**



**Fig 3.6 OUTPUT AS GAME ENDS**

## **CHAPTER 4**

### **SUMMARY AND CONCLUSION**

#### **4.1 SUMMARY**

Just like how Python comes with several modules like random, math, or time that provide additional functions for your programs, the Pygame framework includes several modules with functions for drawing graphics, playing sounds, handling mouse input, and other things

#### **4.2 CONCLUSION**

We conclude that Python is a easy programming language that can be used to develop Games. This project is about creating 2d game called spave invaders in python programming.Perhaps, the most import lesson is that following good software design and coding practices is essential to the success of the game

#### **4.3 FUTURE WORK**

Python has been an important part of Google since the beginning, and remains so as the system grows and evolves. Today, dozens of Google engineers use python, and we're looking for more people with skills in this language.

This language was designed with an aim to provide platform independence and code readability in mind. These 2 unique features make this language logical and adaptable as well as easier to understand. The flexibility that Python runs same source code on a wide array of platforms, which so ever it operates makes it stand out of the crowd in programming market today, where majorly all other languages fall short. Python has been described as most organised and standardized language promises a well built career ahead for all the aspirants.

## APPENDIX

### SOURCE CODE

```
1  import pygame
2  import random
3  import math
4  from pygame import mixer
5
6  # initializing pygame
7  pygame.init()
8
9  # creates the screen
10 screen = pygame.display.set_mode((800, 600))
11
12 # Title and Icon
13 pygame.display.set_caption("Galactic Invaders")
14 icon = pygame.image.load('Space Ship.png')
15 pygame.display.set_icon(icon)
16
17 # Background
18 background = pygame.image.load('Background.png')
19
20
21 # running = True
22 # Background sound
23 mixer.music.load('Background.wav')
24 mixer.music.play(-1)
25
26
27 # Player
28 playerImg = pygame.image.load('Player.png')
29 playerX = 780
30 playerY = 530
31 playerX_change = 0
32
33
34 # Enemy
35 EnemyImg = []
36 EnemyX = []
```

```

37 EnemyY = []
38 EnemyX_change = []
39 EnemyY_change = []
40 num_of_enemies = 6
41 for i in range(num_of_enemies):
42     EnemyImg.append(pygame.image.load('Enemy.png'))
43     EnemyX.append(random.randint(0, 736))
44     EnemyY.append(random.randint(0, 10))
45     EnemyX_change.append(2)
46     EnemyY_change.append(40)
47
48
49 # Bullet
50 # Ready- You cant see the bullet on the screen
51 # Fire- the bullet is moving
52 BulletImg = pygame.image.load('Bullet.png')
53 BulletX = 0
54 BulletY = 530
55 BulletX_change = 0
56 BulletY_change = 20
57 Bullet_state = "ready"
58
59
60 # game over sound
61 GameOver_sound = mixer.Sound('Gameover.wav')
62 game_state = 'ready'
63 # GameOver_sound.play()
64
65
66 # Score
67 score_value = 0
68 font = pygame.font.Font('freesansbold.ttf', 32)
69 textX = 10
70 textY = 10
71
72
73 # Game Over text
74 over_font = pygame.font.Font('freesansbold.ttf', 64)

```

```

75
76
77 def show_score(X, Y):
78     score = font.render("Score : " + str(score_value), True, (255, 255, 255))
79     screen.blit(score, (X, Y))
80
81
82 def game_over_text():
83     over_text = over_font.render("GAME OVER", True, (255, 255, 255))
84     screen.blit(over_text, (200, 250))
85
86
87 # function for player
88 def player(x, y):
89     screen.blit(playerImg, (x, y))
90
91
92 # function for Enemy
93 def Enemy(x, y, i):
94     screen.blit(EnemyImg[i], (x, y))
95
96
97 def fire_bullet(X, y):
98     global Bullet_state
99     Bullet_state = 'fire'
100     screen.blit(BulletImg, (X + 16, y + 10))
101
102
103 def game_over_sound():
104     global game_state
105     game_state = 'sing'
106     mixer.music.stop()
107    GameOver_sound.play()
108
109
110 def iscollision(EnemyX, EnemyY, BulletX, BulletY):
111     distance = math.sqrt((math.pow(BulletX - EnemyX, 2)) + (math.pow(BulletY -
112     EnemyY, 2)))

```

```

113     if distance < 26:
114         return True
115     else:
116         return False
117
118
119 # Game Loop
120 running = True
121 while running:
122     # RGB red green blue
123     screen.fill((0, 0, 0))
124
125     # Background image
126     screen.blit(background, (0, 0))
127
128     for event in pygame.event.get():
129         if event.type == pygame.QUIT:
130             running = False
131
132     # whenever a key is pressed key down is activated
133     if event.type == pygame.KEYDOWN:
134         if event.key == pygame.K_LEFT:
135             playerX_change = -5
136         if event.key == pygame.K_RIGHT:
137             playerX_change = 5
138         if event.key == pygame.K_SPACE:
139             if Bullet_state is 'ready':
140                 Bullet_sound = mixer.Sound('Gunshot.wav')
141                 Bullet_sound.play()
142                 BulletX = playerX
143                 fire_bullet(BulletX, BulletY)
144
145
146     # whenever a key is released key up is activated
147     if event.type == pygame.KEYUP:
148         if event.key == pygame.K_LEFT or event.key == pygame.K_RIGHT:
149             playerX_change = 0
150

```

```

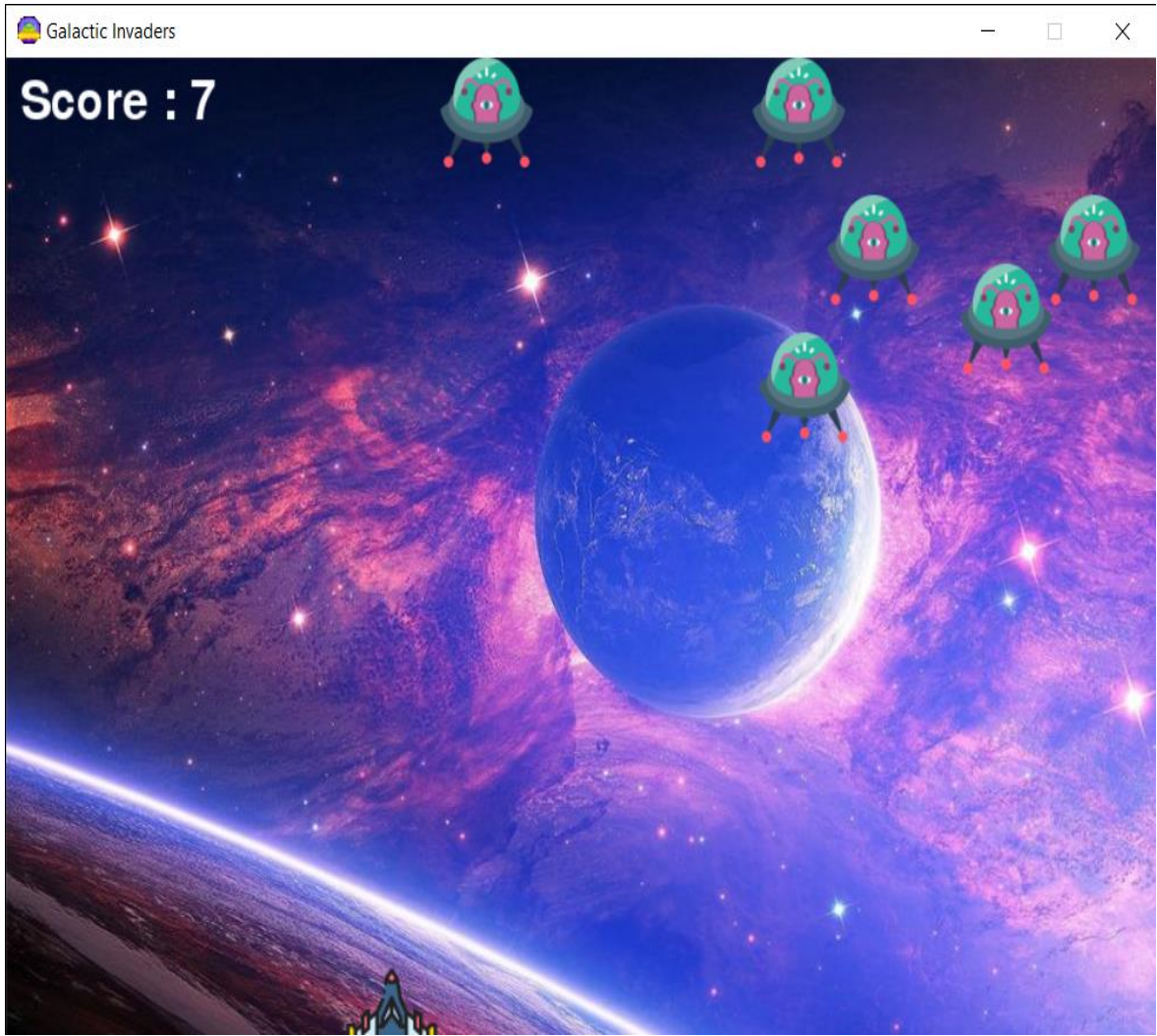
151
152     # S = S + 0.1
153     # S = S + -0.1
154     playerX += playerX_change
155     if playerX <= 0:
156         playerX = 0
157     if playerX >= 736:
158         playerX = 736
159
160
161     # Enemy movements
162     for i in range(num_of_enemies):
163         # Game Over
164         if EnemyY[i] > 450:
165             for j in range(num_of_enemies):
166                 EnemyY[j] = 1000
167                 game_over_text()
168                 break
169
170         EnemyX[i] += EnemyX_change[i]
171         if EnemyX[i] <= 0:
172             EnemyX_change[i] = 2
173             EnemyY[i] += EnemyY_change[i]
174         elif EnemyX[i] >= 736:
175             EnemyX_change[i] = -2
176             EnemyY[i] += EnemyY_change[i]
177         # Collision
178         collision = iscollision(EnemyX[i], EnemyY[i], BulletX, BulletY)
179         if collision:
180             Explosion_sound = mixer.Sound('Explosion.wav')
181             Explosion_sound.play()
182             BulletY = 480
183             Bullet_state = 'ready'
184             score_value += 1
185             EnemyX[i] = random.randint(0, 30)
186             EnemyY[i] = random.randint(0, 0)
187
188         Enemy(EnemyX[i], EnemyY[i], i)

```

```
189
190     # Bullet movements
191     if BulletY <= -20:
192         BulletY = 480
193         Bullet_state = 'ready'
194
195     if Bullet_state is "fire":
196         fire_bullet(BulletX, BulletY)
197         BulletY -= BulletY_change
198
199     # Player function
200     player(playerX, playerY)
201
202     show_score(textX, textY)
```

## OUTPUT

203



204

## REFERENCES

1. Introduction to Programming using Python by Y.Daniel Liang published by Pearson.
2. Python for complete beginners by Dr. Martin Jones published by Createspace Independent Pub on date 18 June 2015.
3. <https://www.startertutorials.com/blog/python-programming-tutorial.html>  
[https://www.tutorialspoint.com/python3/python\\_quick\\_guide.html](https://www.tutorialspoint.com/python3/python_quick_guide.html)
4. EDDY, A. 2003. On with the show... again.  
<http://www.gamespy.com/bizbuzz/january03/bizbuzz49> (January).
5. FOLEY, J., VAN DAM, A., FEINER, S., AND HUGHES, J. 1990. Computer Graphics: Principles and Practice, 2nd Edition. Addison Wesley.
6. FUCHS, H., KEDEM, Z., AND NAYLOR, B. 1980. On visible surface generation by a priori tree structures. In Proceedings of SIGGRAPH, ACM SIGGRAPH / Addison Wesley, Computer Graphics Proceedings, Annual Conference Series, 124–133.