**SPACE SHOOTER**

**INDUSTRIAL TRAINING PROJECT REPORT**

***Submitted in partial fulfilment of th*e *requirements for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

***in***

**INFORMATION TECHNOLOGY**

***by***

**TANVI GULATI(03296203116)**

**ANJALI AGGARWAL(00496204116)**

**CHARCHIKA GUPTA(35296203116)**

**SHIVANGI JAIN(40596203116)**

**SAAD ALI(00396207717)**

**Mr. SUNNY AND Ms. JASNEET**



**DEPARTMENT OF INFORMATION TECHNOLGY**

**DR.AKHILESH DASS GUPTA INSTITUTE OF TECH & MGMT**

**(AFFILIATED TO GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY,**

**DELHI)**

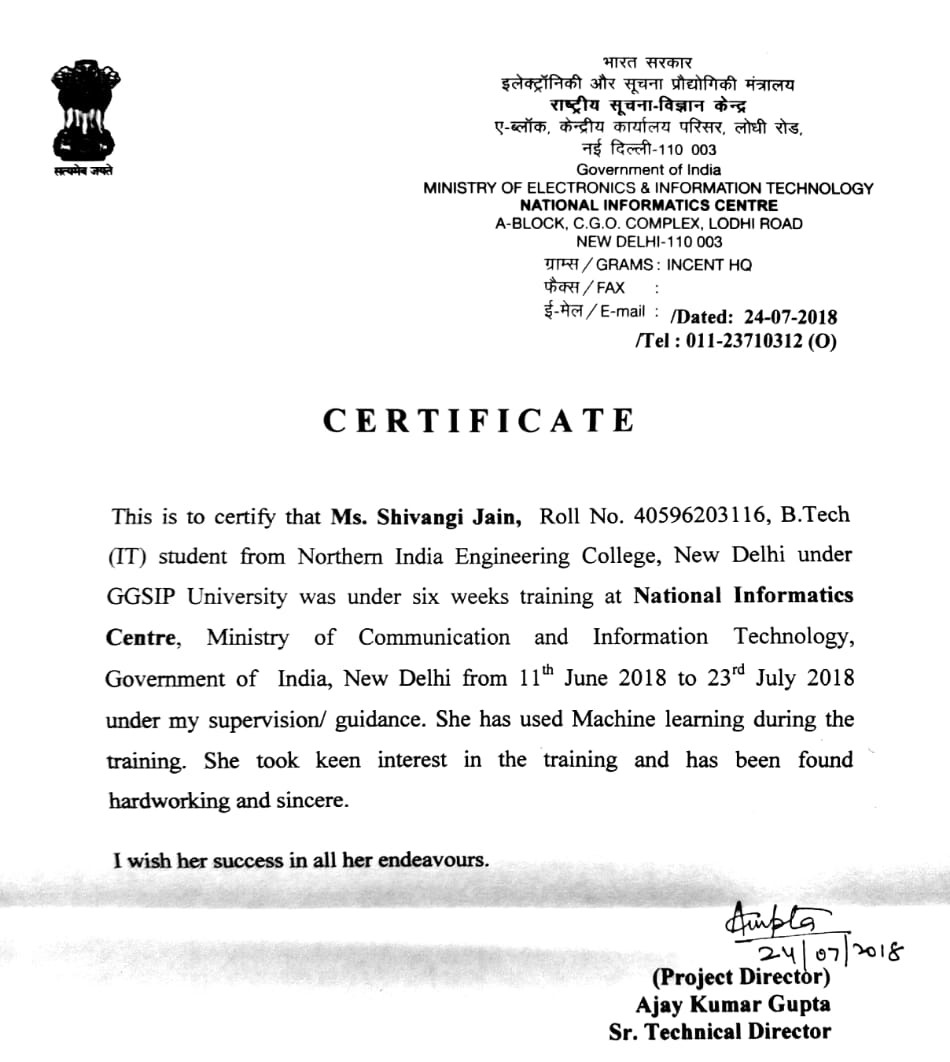
**NEW DELHI – 110053**

**MAY 2019**

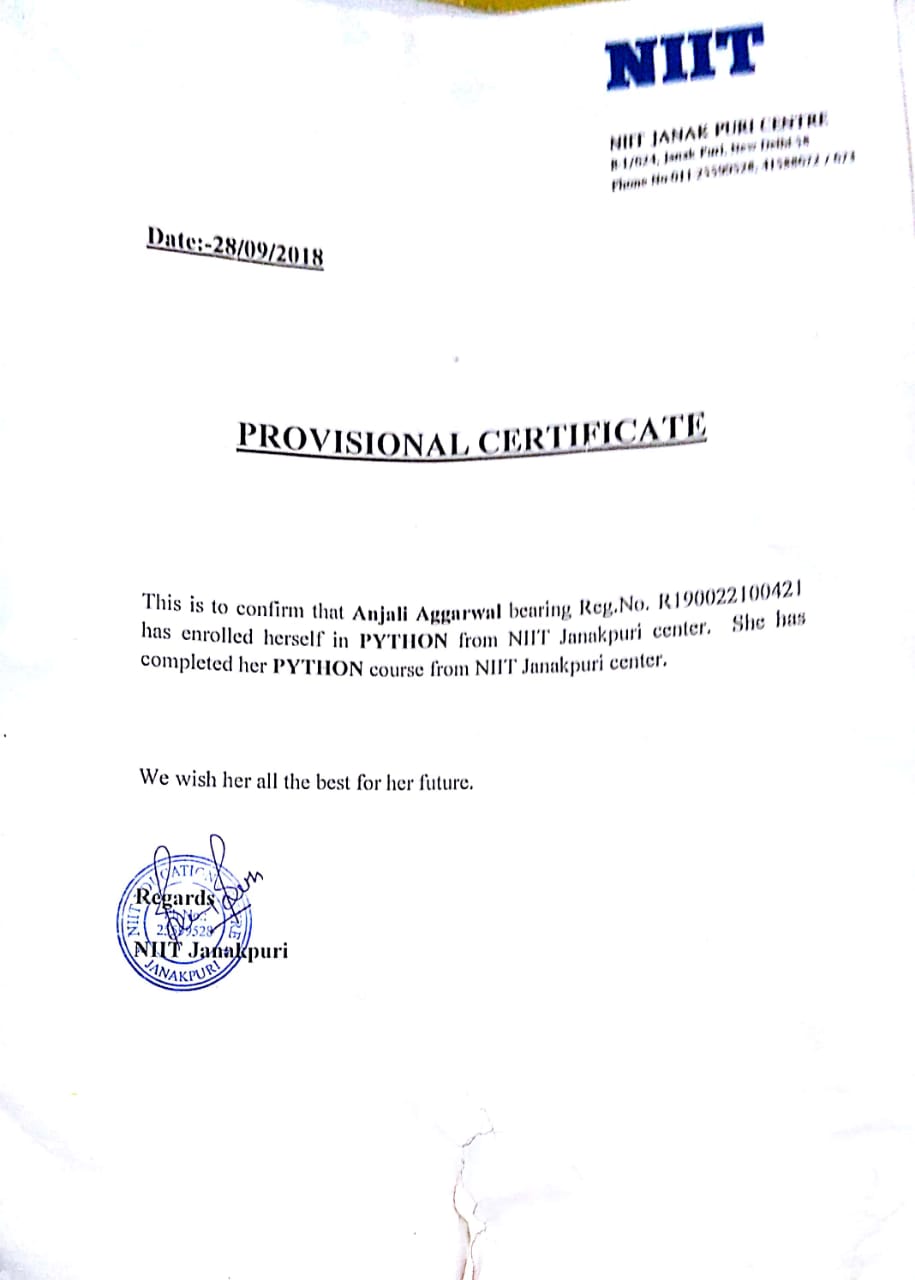
**CERTIFICATE OF TRAINING**

****

****

****

****

****

**ABSTRACT**

**Python** is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) [high-level programming language](https://en.wikipedia.org/wiki/High-level_programming_language) for [general-purpose programming](https://en.wikipedia.org/wiki/General-purpose_programming_language). Created by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) and first released in 1991, Python has a design philosophy that emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability), notably using [significant whitespace](https://en.wikipedia.org/wiki/Significant_whitespace).

Python features a [**dynamic type**](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [**memory management**](https://en.wikipedia.org/wiki/Memory_management) .It supports multiple [**programming paradigms**](https://en.wikipedia.org/wiki/Programming_paradigm) , including [**object-oriented**](https://en.wikipedia.org/wiki/Object-oriented_programming), [**imperative**](https://en.wikipedia.org/wiki/Imperative_programming),  [**functional**](https://en.wikipedia.org/wiki/Functional_programming)and [**procedural**](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive [**standard library**](https://en.wikipedia.org/wiki/Standard_library)**.**

**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.

The game "SPACE SHOOTER " is designed using python. We have used the functions which are included in the pygame module (which is a 3rd party module). Now in this game we have two ships one is the enemy ship and other one is my ship. Both ships are capable of firing bullets. If bullets from myship destroys all the enemyships then player is winner or if bullets from enemyship collide with myship then player has lost. The game runs for a certain period of time and player has to destroy the ships in that time only. Also we have used various images and sounds in this games to make is more user friendly.

**TABLE OF CONTENTS**

**ABSTRACT**

**TABLE OF CONTENTS**

**LIST OF FIGURES**

**LIST OF KEYWORDS**

**LIST OF ABBREVIATIONS**

Chapter 1: Introduction

1.1 What is Python? 8

1.2 How is python used? 9

1.3 Software used 10

1.4 Modules used in python 11

1.5 Python as a dynamic language 12

1.6 Advantages Of python 13-15

1.7 Motivation 16

1.8 Objective 17

Chapter 2: Applications of python

2.1 List of application 18-20

Chapter 3: Pygame Module

3.1 Introduction 21-23

3.2 Basic pygame program 24

Chapter 5: Collision

5.1 What is collision? 25

5.2 Types of collision 25

5.3 Command used for collision 26

Chapter 6: Result Analysis

6.1 Snapshots Of Output 27-28

Chapter 7:Conclusion And Future Scope

7.1Conclusion 29

7.2Future Scope 29

Chapter 8:Biblography 30

**LIST OF FIGURES**

**Figure no Figure Name Page no**

Figure 1 pygame 21

Figure 2 pygame 22

Figure 3 snapshots 27

FIgure 4 snapshots 27

Figure 5 snapshots 28

**LIST OF KEYWORDS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [False](https://www.programiz.com/python-programming/keyword-list#true_false) | [class](https://www.programiz.com/python-programming/keyword-list#class) | [finally](https://www.programiz.com/python-programming/keyword-list#finally) | [is](https://www.programiz.com/python-programming/keyword-list#is) | [return](https://www.programiz.com/python-programming/keyword-list#return) |
| [None](https://www.programiz.com/python-programming/keyword-list#none) | [continue](https://www.programiz.com/python-programming/keyword-list#break_continue) | [for](https://www.programiz.com/python-programming/keyword-list#for) | [lambda](https://www.programiz.com/python-programming/keyword-list#lambda) | [try](https://www.programiz.com/python-programming/keyword-list#except_raise_try) |
| [True](https://www.programiz.com/python-programming/keyword-list#true_false) | [def](https://www.programiz.com/python-programming/keyword-list#def) | [from](https://www.programiz.com/python-programming/keyword-list#from_import) | [nonlocal](https://www.programiz.com/python-programming/keyword-list#nonlocal) | [while](https://www.programiz.com/python-programming/keyword-list#while) |
| [and](https://www.programiz.com/python-programming/keyword-list#and_or_not) | [del](https://www.programiz.com/python-programming/keyword-list#del) | [global](https://www.programiz.com/python-programming/keyword-list#global) | [not](https://www.programiz.com/python-programming/keyword-list#and_or_not) | [with](https://www.programiz.com/python-programming/keyword-list#with) |
| [as](https://www.programiz.com/python-programming/keyword-list#as) | [elif](https://www.programiz.com/python-programming/keyword-list#if_else_elif) | [if](https://www.programiz.com/python-programming/keyword-list#if_else_elif) | [or](https://www.programiz.com/python-programming/keyword-list#and_or_not) | [yield](https://www.programiz.com/python-programming/keyword-list#yield) |
| [assert](https://www.programiz.com/python-programming/keyword-list#assert) | [else](https://www.programiz.com/python-programming/keyword-list#if_else_elif) | [import](https://www.programiz.com/python-programming/keyword-list#from_import) | [pass](https://www.programiz.com/python-programming/keyword-list#pass) |  |
| [break](https://www.programiz.com/python-programming/keyword-list#break_continue) | [except](https://www.programiz.com/python-programming/keyword-list#except_raise_try) | [in](https://www.programiz.com/python-programming/keyword-list#in) | [raise](https://www.programiz.com/python-programming/keyword-list#except_raise_try) |  |

**CHAPTER 1: INTRODUCTION**

**1.1What Is Python?**

In technical terms, Python is an object-oriented, high-level programming language with integrated dynamic semantics primarily for web and app development. It is extremely attractive in the field of Rapid Application Development because it offers dynamic typing and dynamic binding options.

Python is relatively simple, so it's easy to learn since it requires a unique syntax that focuses on readability. Developers can read and translate Python code much easier than other languages. In turn, this reduces the cost of program maintenance and development because it allows teams to work collaboratively without significant language and experience barriers.

Additionally, Python supports the use of modules and packages, which means that programs can be designed in a modular style and code can be reused across a variety of projects. Once you've developed a module or package you need, it can be scaled for use in other projects, and it's easy to import or export these modules.

One of the most promising benefits of Python is that both the standard library and the interpreter are available free of charge, in both binary and source form. There is no exclusivity either, as Python and all the necessary tools are available on all major platforms. Therefore, it is an enticing option for developers who don't want to worry about paying high development costs.

If this description of Python over your head, don't worry. You'll understand it soon enough. What you need to take away from this section is that Python is a programming language used to develop software on the web and in app form, including mobile.

**1.2 How Is Python Used?**

Python is a general-purpose programming language, which is another way to say that it can be used for nearly everything. Most importantly, it is an interpreted language, which means that the written code is not actually translated to a computer-readable format at runtime. Whereas, most programming languages do this conversion before the program is even run. This type of language is also referred to as a "scripting language" because it was initially meant to be used for trivial projects.

The concept of a "scripting language" has changed considerably since its inception, because Python is now used to write large, commercial style applications, instead of just banal ones. This reliance on Python has grown even more so as the internet gained popularity. A large majority of web applications and platforms rely on Python, including Google's search engine, YouTube, and the web-oriented transaction system of the New York Stock Exchange (NYSE). You know the language must be pretty serious when it's powering a stock exchange system.

In fact, NASA actually uses Python when they are programming their equipment and space machinery.

Python can also be used to process text, display numbers or images, solve scientific equations, and save data. In short, it is used behind the scenes to process a lot of elements you might need or encounter on your device(s) - mobile included.

**1.3 Software Used**

**Pycharm**

**PyCharm** is an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) used in [computer programming](https://en.wikipedia.org/wiki/Computer_programming), specifically for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) language. It is developed by the Czech company [JetBrains](https://en.wikipedia.org/wiki/JetBrains).[[2]](https://en.wikipedia.org/wiki/PyCharm#cite_note-2) It provides code analysis, a graphical debugger, an integrated unit tester, integration with [version control systems](https://en.wikipedia.org/wiki/Revision_control) (VCSes), and supports web development with [Django](https://en.wikipedia.org/wiki/Django_(web_framework)).

PyCharm is [cross-platform](https://en.wikipedia.org/wiki/Cross-platform), with [Windows](https://en.wikipedia.org/wiki/Windows), [macOS](https://en.wikipedia.org/wiki/MacOS) and [Linux](https://en.wikipedia.org/wiki/Linux) versions. The Community Edition is released under the [Apache License](https://en.wikipedia.org/wiki/Apache_License),[[3]](https://en.wikipedia.org/wiki/PyCharm#cite_note-community-3) and there is also Professional Edition released under a [proprietary license](https://en.wikipedia.org/wiki/Proprietary_software) - this has extra features.

**Features of pycharm**

* Coding assistance and [analysis](https://en.wikipedia.org/wiki/Code_analysis), with [code completion](https://en.wikipedia.org/wiki/Autocomplete), syntax and error highlighting, linter integration, and quick fixes
* Project and code navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages
* Python [refactoring](https://en.wikipedia.org/wiki/Refactoring): including rename, extract method, introduce variable, introduce constant, pull up, push down and others
* Support for web frameworks: [Django](https://en.wikipedia.org/wiki/Django_(web_framework)), [web2py](https://en.wikipedia.org/wiki/Web2py) and [Flask](https://en.wikipedia.org/wiki/Flask_(web_framework))
* Integrated Python [debugger](https://en.wikipedia.org/wiki/Debugger)
* Integrated [unit testing](https://en.wikipedia.org/wiki/Unit_testing), with line-by-line [code coverage](https://en.wikipedia.org/wiki/Code_coverage)
* [Google App Engine](https://en.wikipedia.org/wiki/Google_App_Engine) Python development
* Version control integration: unified user interface for [Mercurial](https://en.wikipedia.org/wiki/Mercurial), [Git](https://en.wikipedia.org/wiki/Git_(software)), [Subversion](https://en.wikipedia.org/wiki/Apache_Subversion), [Perforce](https://en.wikipedia.org/wiki/Perforce) and [CVS](https://en.wikipedia.org/wiki/Concurrent_Versions_System) with changelists and merge

**1.4 Modules Used In Python**

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.

Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.

The  Import  Statement

You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* has the following syntax −

import module1[,module2 --------------- module n]

Types Of Modules

1. Built-In : These modules are present in python. eg - random.

2. User-Defined : These modules are defined by user.

3. 3rd Party : These modules are not defined instead are installed.

**1.5 Python As A Dynamic Language**

Dynamic programming language, in computer science, is a class of high-level programming languages which, at run time, execute many common programming behaviour that static programming languages perform during compilation. These behaviour could include extension of the program, by adding new code, by extending objects and definitions, or by modifying the type system. Although similar behaviours can be emulated in nearly any language, with varying degrees of difficulty, complexity and performance costs, dynamic languages provide direct tools to make use of them.

Python is strongly typed as the interpreter keeps track of all variables types. It's also very dynamic as it rarely uses what it knows to limit variable usage. In Python, it's the program's responsibility to use built-in functions like isinstance() and issubclass() to test variable types and correct usage. Python tries to stay out of your way while giving you all you need to implement strong type checking.

**1.6 Advantages Of Python**

**Easy Syntax**

Python's syntax is easy to learn, so both non-programmers and programmers can start programming right away.

**Readability**

Python's syntax is very clear, so it is easy to understand program code. (Python is often referred to as "executable pseudo-code" because its syntax mostly follows the conventions used by programmers to outline their ideas without the formal verbosity of code in most programming languages; in other words syntax of Python is almost identical to the simplified "pseudo-code" used by many programmers to prototype and describe their solution to other programmers. Thus Python can be used to prototype and test code which is later to be implemented in other programming languages).

**High-Level Language**

Python looks more like a readable, human language than like a low-level language. This gives you the ability to program at a faster rate than a low-level language will allow you.

**Object oriented programming**

Object-oriented programming allows you to create data structures that can be re-used, which reduces the amount of repetitive work that you'll need to do. Programming languages usually define objects with namespaces, like class or def, and objects can edit themselves by using keyword, like this or self. Most modern programming languages are object-oriented (such as Java, C++, and C#) or have support for OOP features (such as Perl version 5 and later). Additionally object-oriented techniques can be used in the design of almost any non-trivial software and implemented in almost any programming or scripting language. (For example a number of Linux kernel features are "objects" which implement their own [encapsulation](https://en.wikipedia.org/wiki/Encapsulation_(object-oriented_programming)) of behavior and data structive via pointers, specifically pointers to functions, in the C programming language).[[*citation needed*](https://en.wikiversity.org/wiki/Wikiversity:Citation_needed)] Python's support for object-oriented programming is one of its greatest benefits to new programmers because they will be encountering the same concepts and terminology in their work environment. If you ever decide to switch languages, or use any other for that fact, you'll have a significant chance that you'll be working with object-oriented programming.[[1]](https://en.wikiversity.org/wiki/Python_Concepts/Why_learn_Python#cite_note-1)

**It's Free**

Python is both free and open-source. The Python Software Foundation distributes pre-made binaries that are freely available for use on all major operating systems called CPython. You can get CPython's source-code, too. Plus, you can modify the source code and distribute as allowed by CPython's license. [[2]](https://en.wikiversity.org/wiki/Python_Concepts/Why_learn_Python#cite_note-2) (Luckily, CPython has a [permissive free software license](https://en.wikipedia.org/wiki/Permissive_free_software_licence) attitude.)

**Cross-platform**

Python runs on all major operating systems like Microsoft Windows, Linux, and Mac OS X.

**Widely Supported**

Python has an active support community with many web sites, mailing lists, and USENET "netnews" groups that attract a large number of knowledgeable and helpful contributes.

**It's Safe**

Python doesn't have pointers like other C-based languages, making it much more reliable. Along with that, errors never pass silently unless they're explicitly silenced. This allows you to see and read why the program crashed and where to correct your error.

**Batteries Included**

Python is famous for being the "batteries are included" language.[[3]](https://en.wikiversity.org/wiki/Python_Concepts/Why_learn_Python#cite_note-3) There are over 300 standard library modules which contain modules and classes for a wide variety of programming tasks.[[*citation needed*](https://en.wikiversity.org/wiki/Wikiversity:Citation_needed)] For example the standard library contains modules for safely creating temporary files (named or anonymous), mapping files into memory (including use of shared and anonymous memory mappings), spawning and controlling sub-processes, compressing and decompressing files (compatible with gzip or PK-zip) and archives files (such as Unix/Linux "tar"), accessing indexed "DBM" (database) files, interfacing to various graphical user interfaces (such as the TK toolkit and the popular WxWindows multi-platform windowing system), parsing and maintaining CSV (comma-separated values) and ".cfg" or ".ini" configuration files (similar in syntax to the venerable WIN.INI files from MS-DOS and MS-Windows), for sending e-mail, fetching and parsing web pages, etc. It's possible, for example, to create a custom web server in Python using less than a dozen lines of code, and one of the standard libraries, of course.

**Extensible**

In addition to the standard libraries there are extensive collections of freely available add-on modules, libraries, frameworks, and tool-kits. These generally conform to similar standards and conventions; for example almost all of the database adapters (to talk to almost any client-server RDBMS engine such as MySQL, Postgres, Oracle, etc) conform to the Python DBAPI and thus can mostly be accessed using the same code. So it's usually easy to modify a Python program to support any database engine.

**1.7 Motivation**

The motivation behind building this project was to challenge and further improve my knowledge for Python language. I wanted to create something useful and enjoyable at the same time and hence, created a game space shooter.

**1.8 Objective**

* To understand why Python is a useful scripting language for developers.
* To learn how to design and program Python applications.
* To learn how to use lists, tuples, and dictionaries in Python programs.
* To learn how to identify Python object types.
* To learn how to use indexing and slicing to access data in Python programs.
* To define the structure and components of a Python program.
* To learn how to write loops and decision statements in Python.
* To learn how to write functions and pass arguments in Python.
* To learn how to build and package Python modules for reusability.

**Chapter 2 : Applications Of Python**

**2.1 List Of Applications**

# **Web and Internet Development**

Python offers many choices for [web development](http://wiki.python.org/moin/WebProgramming):

* Frameworks such as [Django](http://www.djangoproject.com/) and [Pyramid](http://www.pylonsproject.org/).
* Micro-frameworks such as [Flask](http://flask.pocoo.org/) and [Bottle](http://bottlepy.org/).
* Advanced content management systems such as [Plone](http://www.plone.org/) and [django CMS](https://www.django-cms.org/).

Python's standard library supports many Internet protocols:

* [HTML and XML](http://docs.python.org/library/markup)
* [JSON](http://docs.python.org/library/json.html)
* [E-mail processing](http://docs.python.org/library/email).
* Support for [FTP](http://docs.python.org/library/ftplib.html), [IMAP](http://docs.python.org/2/library/imaplib.html), and other [Internet protocols](http://docs.python.org/library/internet).
* Easy-to-use [socket interface](http://docs.python.org/howto/sockets.html).

And the Package Index has yet more libraries:

* [Requests](https://pypi.python.org/pypi/requests/), a powerful HTTP client library.
* [BeautifulSoup](http://www.crummy.com/software/BeautifulSoup/), an HTML parser that can handle all sorts of oddball HTML.
* [Feedparser](https://pypi.python.org/pypi/feedparser) for parsing RSS/Atom feeds.
* [Paramiko](https://pypi.python.org/pypi/paramiko/), implementing the SSH2 protocol.
* [Twisted Python](http://twistedmatrix.com/), a framework for asynchronous network programming.

# **Scientific and Numeric**

Python is widely used in [scientific and numeric](http://wiki.python.org/moin/NumericAndScientific) computing:

* [SciPy](http://scipy.org/) is a collection of packages for mathematics, science, and engineering.
* [Pandas](http://pandas.pydata.org/) is a data analysis and modeling library.
* [IPython](http://ipython.org/) is a powerful interactive shell that features easy editing and recording of a work session, and supports visualizations and parallel computing.
* The [Software Carpentry Course](http://software-carpentry.org/) teaches basic skills for scientific computing, running bootcamps 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.

* Books such as [How to Think Like a Computer Scientist](http://www.openbookproject.net/thinkcs/python/english2e/), [Python Programming: An Introduction to Computer Science](http://mcsp.wartburg.edu/zelle/python/), and [Practical Programming](http://pragprog.com/book/gwpy2/practical-programming).
* The [Education Special Interest Group](https://www.python.org/community/sigs/current/edu-sig) is a good place to discuss teaching issues.

# **Desktop GUIs**

The [Tk](http://wiki.python.org/moin/TkInter) GUI library is included with most binary distributions of Python.

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

* [wxWidgets](http://www.wxpython.org/)
* [Kivy](http://kivy.org/), for writing multitouch applications.
* Qt via [pyqt](http://www.riverbankcomputing.co.uk/software/pyqt/intro) or [pyside](http://www.pyside.org/)

Platform-specific toolkits are also available:

* [GTK+](http://www.pygtk.org/)
* Microsoft Foundation Classes through the [win32 extensions](http://sourceforge.net/projects/pywin32/)

# **Software Development**

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

* [SCons](http://www.scons.org/) for build control.
* [Buildbot](http://buildbot.sourceforge.net/) and [Apache Gump](http://gump.apache.org/) for automated continuous compilation and testing.
* [Roundup](http://roundup.sourceforge.net/) or [Trac](http://www.edgewall.com/trac/) for bug tracking and project management.

# **Business Applications**

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

* [Odoo](https://www.odoo.com/) is an all-in-one management software that offers a range of business applications that form a complete suite of enterprise management applications.
* [Tryton](http://www.tryton.org/) is a three-tier high-level general purpose application platform.

**Chapter 3 : Pygame Module**

**3.1 Introduction**

"Is Python suitable for gaming?" The answer is, "It depends on the game."

Python is actually quite capable at running games. It will likely even surprise you how much is possible in under 30 milliseconds. Still, it is not hard to reach the ceiling once your game begins to get more complex. Any game running in realtime will be making full use of the computer.



Figure 1

Over the past several years there has been an interesting trend in game development, the move towards higher level languages. Usually a game is split into two major parts. The game engine, which must be as fast as possible, and the game logic, which makes the engine actually do something. It wasn't long ago when the engine of a game was written in assembly, with portions written in C. Nowadays, C has moved to the game engine, while often the game itself is written in higher level scripting languages. Games like Quake3 and Unreal run these scripts as portable bytecode.

In early 2001, developer Rebel Act Studios finished their game, Severance: Blade of Darkness. Using their own custom 3D engine, the rest of the game is written with Python. The game is a bloody action 3rd person perspective fighter. You control medieval warriors into intricate decapitating combination attacks while exploring dungeons and castles. You can download third party add-ons for this game, and find they are nothing more than Python source files.

More recently, Python has been used in a variety of games like Freedom Force, and Humungous' Backyard Sports Series.



Figure 2

Pygame serve as an excellent C engine for 2D games. Games will still find the largest part of their runtime is spent inside SDL handling the graphics. SDL can take advantage of graphics hardware acceleration. Enabling this can change a game from running around 40 frames per second to over 200 frames per second. When you see your Python game running at 200 frames per second, you realize that Python and games can work together.

It is impressive how well Python work on multiple platforms. For example, in May of 2001 I released my own full pygame project, SolarWolf, an arcade style action game. One thing that has surprised me is that one year later there has been no need for any patches, bug fixes, or updates. The game was developed entirely on windows, but runs on Linux, Mac OSX, and many Unixes without any extra work on my end.

Pygame is fairly low-level when it comes to writing games. You'll quickly find yourself needing to wrap common functions into your own game environment. The great thing about this is there is nothing inside pygame to get in your way. Your program is in full control of everything. The side effect of that is you will find yourself borrowing a lot of code to get a more advanced framework put together. You'll need a better understanding of what you are doing.

**3.2 Basic Pygame Program**

First step is always to import the pygame module using command:

Import pygame

Pygame.init() is used to initialise pygame using init() function.

#basic program

import pygame

pygame.init()

width = 1000

height = 500

# 0-255

# black = 0,0,0

# white = 255,255,255

red = 255,0,0

screen = pygame.display.set\_mode((width, height))

screen.fill(red)

while True:

for event in pygame.event.get():

# print(event)

if event.type == pygame.QUIT:

pygame.quit() # quit pygame

quit() # quit python

pygame.display.update()

**Chapter 4 : Collision**

**4.1What Is CollisIon?**

Collision detection is figuring when two things on the screen have touched (that is, collided with) each other. For example, if the player touches an enemy they may lose health. Or the program needs to know when the player touches a coin so that they automatically pick it up. Collision detection can help determine if the game character is standing on solid ground or if there’s nothing but empty air underneath them.

**4.2 Types Of Collision**

There are mainly two types of collision in python:

**1.Rectangle collision**

In this type of collision occurs as soon as the rectangles behind the image come in contact with each other.

**2.Pixel Collision**

Instead of using 8-32bits per pixel, pygames masks use only 1 bit per pixel. This makes it very quick to check for collisions. As you can compare 32 pixels with one integer compare. Masks use bounding box collision first - to speed things up.   
Even though bounding boxes are a crude approximation for collisions, they are faster than using bitmasks. So pygame first does a check to see if the rectangles collide - then if the rectangles do collide, only then does it check to see if the pixels collide.

**4.3Command Used For Rectangle Collision**

In this project we have used the concept of rectangle collision.

The following syntax is used for collision:

rect1 = pygame.Rect(x-coordinate,y-coordinate,length,breadth)

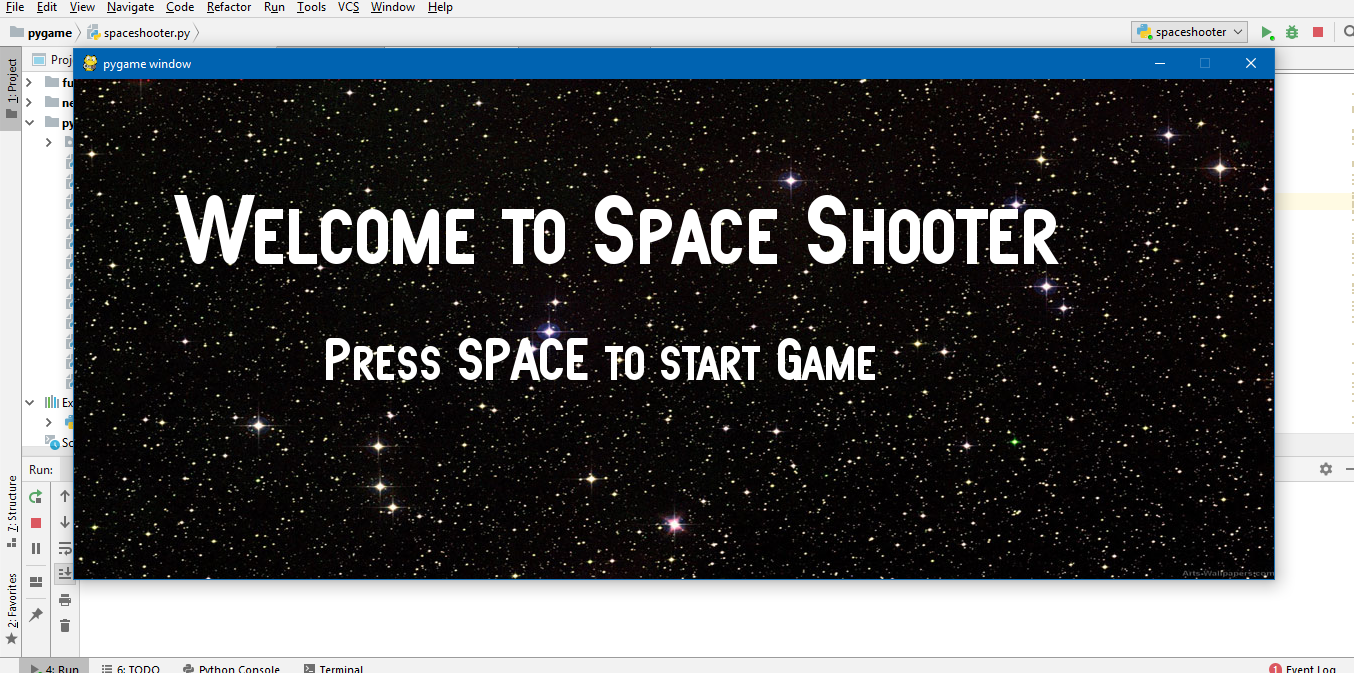
rect2 = pygame.Rect(x-coordinate,y-coordinate,length,breadth)

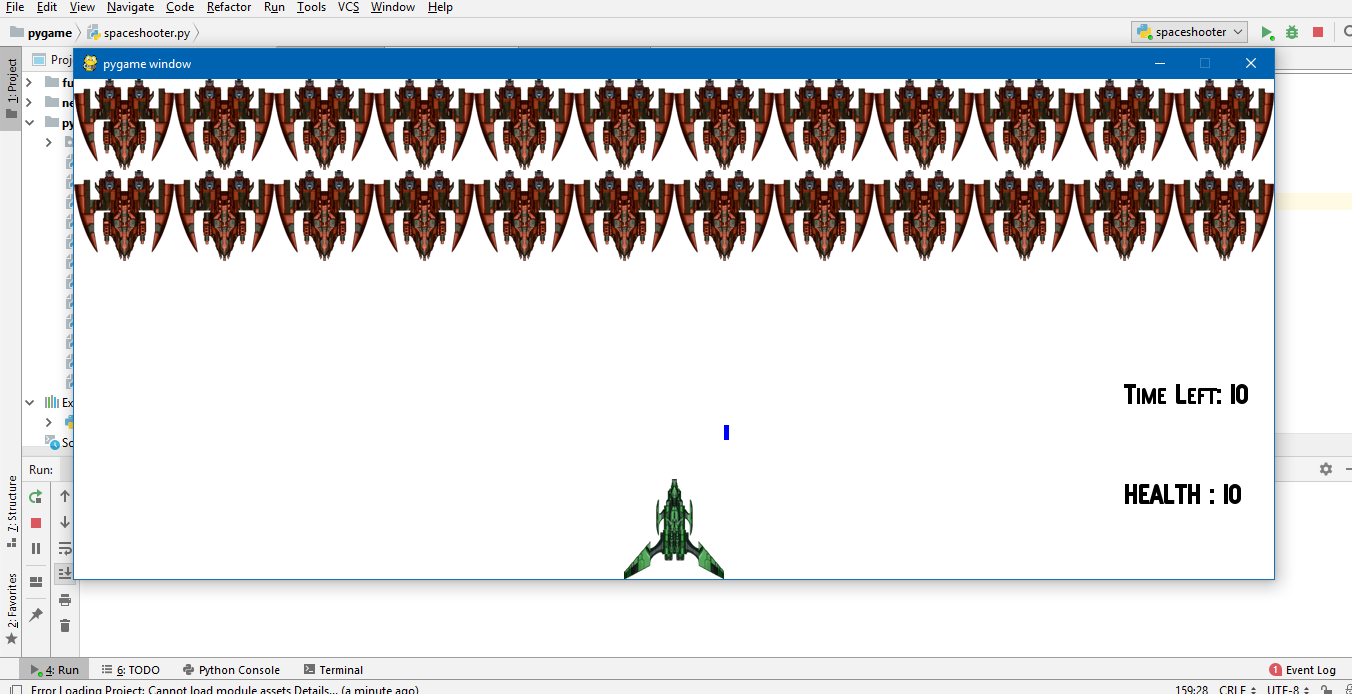
if rect1.colliderect(rect2):

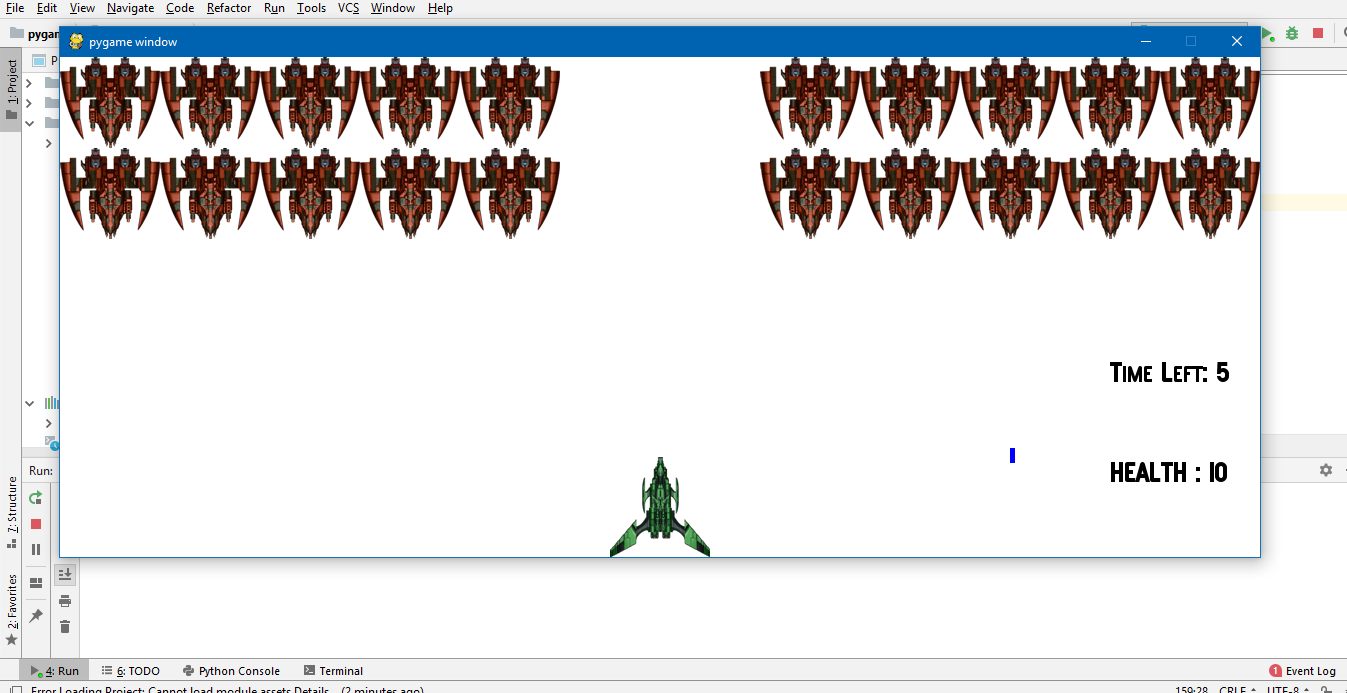
#body

**CHAPTER 5 :RESULT ANALYSIS**

**5.1 Snapshots of program**







**CHAPTER 6 : CONCLUSION AND FUTURE SCOPE**

**6.1 Conclusion**

Python is considered to be the most reliable tool for developing application . Efficient exploration and implementing of such functionalities of python has lead to practical challenging research activities and development of new standards , thereby shrinking the communication gap.

The game works perfectly and can be played by two persons. It shows accurate marking and results and aids user in playing a game of reverse following all rules.

**6.2 Future Scope**

The current program is based on core python which does not include all the functionalities unlike advance python which supports more functionalities . By use of advance python we can improve the UI of this game.

**CHAPTER 7 : BIBLOGRAPHY**

* **Head First Python by Paul Barry**
* **Geeks for Geeks**
* **Stack Overflow**
* **Google.com**
* **Wikipedia**
* **Youtube.com**