Project Report

Industrial Training

M Code: 78336

DAV Institute of Engineering & Technology, Jalandhar

Department of Computer Science and Engineering

Submitted by:

Radhika Sharma

1803689

Date: 20/1/2021

Index

1. Introduction

1.1 What Is Python

1.2 History

1.3 Python 2 Vs Python 3

1.4 Features of Python

2. Python Applications

2.1 **Web Development**

2.2 **Game Development**

2.3 **Machine Learning and Artificial Intelligence**

2.4 **Data Science and Data Visualization**

2.5 **Desktop GUI**

2.6 **Business Applications**

3. IDE It’s Types and Concepts

3.1 Basic Syntax

3.2 Libraries

3.3 Data Structure

3.4 Sets, Dictionaries, Tuple and Lists

3.5 File Handling, Modules and Exceptions.

3.6 Oops

4. Assignments/Projects

4.1 Assignment on Leap Year

4.2 Assignment on Try and Excep

4.3 Assignment on Timer

4.4Project on Hangman Game

5. Conclusion and Future Work

5.1 Expertise in Core Python

5.2 Good Grasp of Web Frameworks

5.3 Road to Data Science

5.4Machine Learning And AI

5.5 Analytical Skills

5.6 Design Skills

5.7 Communication Skills

**1. INTRODUCTION**

1.1 What is Python

Python is a general purpose, dynamic, [high-level](https://www.javatpoint.com/classification-of-programming-languages), and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.Python's syntax and *dynamic typing* with its interpreted nature make it an ideal language for scripting and rapid application development.

Python is not intended to work in a particular area, such as web programming. That is why it is known as *multipurpose* programming language because it can be used with web, enterprise, 3D CAD, etc.Python makes the development and debugging *fast* because there is no compilation step included in Python development, and edit-test-debug cycle is very fast.

1.2 Python History

Python was invented by **Guido van Rossum** in 1991 at CWI in Netherland. The idea of Python programming language has taken from the ABC programming language or we can say that ABC is a predecessor of Python language.

There is also a fact behind the choosing name Python. Guido van Rossum was a fan of the popular BBC comedy show of that time, **"Monty Python's Flying Circus"**. So he decided to pick the name **Python** for his newly created programming language.

Python has the vast community across the world and releases its version within the short period.

1.3 Python 2 vs. Python 3

In most of the programming languages, whenever a new version releases, it supports the features and syntax of the existing version of the language, therefore, it is easier for the projects to switch in the newer version. However, in the case of Python, the two versions Python 2 and Python 3 are very much different from each other.

A list of differences between Python 2 and Python 3 are given below:

1. Python 2 uses **print** as a statement and used as print "something" to print some string on the console. On the other hand, Python 3 uses **print** as a function and used as print("something") to print something on the console.
2. Python 2 uses the function raw\_input() to accept the user's input. It returns the string representing the value, which is typed by the user. To convert it into the integer, we need to use the int() function in Python. On the other hand, Python 3 uses input() function which automatically interpreted the type of input entered by the user. However, we can cast this value to any type by using primitive functions (int(), str(), etc.).
3. In Python 2, the implicit string type is ASCII, whereas, in Python 3, the implicit string type is Unicode.
4. Python 3 doesn't contain the xrange() function of Python 2. The xrange() is the variant of range() function which returns a xrange object that works similar to Java iterator. The range() returns a list for example the function range(0,3) contains 0, 1, 2.
5. There is also a small change made in Exception handling in Python 3. It defines a keyword **as** which is necessary to be used. We will discuss it in Exception handling section of Python programming tutorial.

## 1.4 Features of Python

Python provides many useful features to the programmer. These features make it most popular and widely used language. We have listed below few-essential feature of Python.

* Easy to use and Learn
* Expressive Language
* Interpreted Language
* Object-Oriented Language
* Open Source Language
* Extensible
* Learn Standard Library
* GUI Programming Support
* Integrated
* Embeddable
* Dynamic Memory Allocation
* Wide Range of Libraries and Frameworks

**2. Python Applications**

Python supports **cross-platform operating systems** which makes building applications with it all the more convenient. Some of the globally known applications such as [YouTube](https://www.youtube.com/user/edurekaIN?sub_confirmation=1), BitTorrent, DropBox, etc. use Python to achieve their functionality.

2.1.**Web Development**

Python can be used to make [web-applications](https://www.edureka.co/blog/django-tutorial/) at a rapid rate .It is because of the frameworks Python uses to create these applications. There is common-backend logic that goes into making these frameworks and a number of libraries that can help integrate protocols such as HTTPS, FTP, SSL etc. and even help in the processing of JSON, XML, E-Mail and so much more.



Some of the most well-known frameworks are [Django](https://www.edureka.co/blog/django-tutorial/), [Flask](https://www.youtube.com/watch?v=lj4I_CvBnt0), Pyramid. The security, scalability, convenience that they provide is commendable if we compare it to starting the development of a website from scratch.

### **2.2. Game Development**

Python is also used in the development of [interactive games](https://www.edureka.co/blog/python-turtle-module/). There are libraries such as PySoy which is a 3D game engine supporting Python 3, [PyGame](https://www.edureka.co/blog/pygame-tutorial" \t "_blank) which provides functionality and a library for game development.



2.**3. Machine Learning and Artificial Intelligence**

[Machine Learning](https://www.edureka.co/blog/machine-learning-career/) and [Artificial Intelligence](https://www.edureka.co/blog/pros-and-cons-of-ai/) are the talks of the town as they yield the most promising careers for the future. We make the computer learn based on past experiences through the data stored or better yet, create algorithms which makes the computer learn by itself.



### **2.4. Data Science and Data Visualization**

Data is money if you know how to extract relevant information which can help you take calculated risks and increase profits. You study the data you have, perform operations and extract the information required. [Libraries](https://www.edureka.co/blog/python-libraries/) such as Pandas, NumPy help you in extracting information.

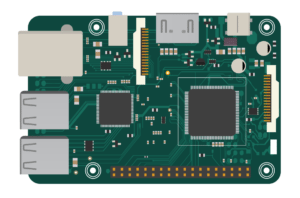


### **2.5. Desktop GUI**

We use Python to program **desktop applications**. It provides the [Tkinter](https://www.edureka.co/blog/tkinter-tutorial/" \t "_blank) library that can be used to develop user interfaces. There are some other useful toolkits such as the wxWidgets, [Kivy](https://www.edureka.co/blog/kivy-tutorial/" \t "_blank), PYQT that can be used to create applications on several platforms.

### **2.6. Business Applications**

Business Applications are different than our normal applications covering domains such as e-commerce, ERP and many more. They require applications which are scalable, extensible and easily readable and Python provides us with all these features. Platforms such as Tryton is available to develop such business applications.



The most well-known embedded application could be the [Raspberry Pi](https://www.edureka.co/blog/raspberry-pi-tutorial/) which uses Python for its computing. We can also use it as a computer or like a simple embedded board to perform high-level computations.

## ****3.IDE it’s types and concepts****

IDE stands for Integrated Development Environment. It is a GUI( Graphical User Interface) where programmers write their code and produce the final products. An IDE basically unifies all essential tools required for software development and testing, which in turn helps the programmer maximize his output. Some IDEs are generic, meaning they can support a number of languages. For example, Sublime Text, Atom, Visual Studio, etc. Language-specific IDEs support a specific language. They also help you understand when you make syntax errors. Example: Pycharm for [Python](https://www.edureka.co/blog/pycharm-tutorial), Jcreator for [Java](https://www.edureka.co/blog/java-tutorial/), RubyMine for [Ruby/Rails](https://www.edureka.co/blog/ruby-on-rails-tutorial/).

## **Difference between IDEs and Code Editors:**

An IDE is a complete environment where you can write, compile, debug, or test your code. On the other hand, Code editors or text editors are platforms where you can just write your code.  The only capacity a code editor should support is editing text. An IDE itself consists of a code editor within its toolkit.

## **Features of an IDE:**

A general IDE should consist of the following:

* **Code Editor**: A code editor is provided to write and manipulate the source code. Code editors can either be standalone applications or integrated into IDEs.
* **Syntax Highlighting:**This feature is provided to mark the syntax of the base language in different colors and fonts.
* **Auto-completion Code:**Designed to minimize time consumption, the auto-completion feature completes or suggests the programmer as to what variables, arguments or code bits need to appear.
* **Debugger:**A debugger is a tool that is required to test and debug the source code.
* **Compiler:**A compiler is a component that translates the source code from one language to another. Compilers usually perform pre-processing, lexical analysis, code optimization, and code generation tasks.
* **Language Support:**IDEs can either be language specific or may have support to multiple languages. The choice relies on the user to single-out and embraces the IDE of his choice.

## **Commonly used IDEs for Python**

Some of the best IDEs for Python are:

### **PyCharm:**

### PyCharm logo-The best IDE for Python-Edureka

Developed by Czech company JetBrains, PyCharm is an IDE specific to Python. PyCharm is a **cross-platform**IDE. Therefore, users may download any of the Windows, Mac or Linux versions depending on their requirements. Honestly, [PyCharm](https://www.edureka.co/blog/pycharm-tutorial) is considered to be one of the best IDE for Python and justifiably the most widely used.

In addition to the common features, PyCharm provides additional features like:

* Specialized project views allowing quick switching between files
* Facilitates Web Development along with [Django](https://www.edureka.co/blog/django-tutorial), Flask, and web2py
* PyCharm is equipped with more than 1000 plug-ins, so programmers can write their own plug-ins to extend its features
* It provides two versions for download, the Community version which is free and the paid Professional version. Programmers can download the corresponding versions as per their requirements

### **IDLE:**

### IDLE logo-The best IDE for Python-Edureka

IDLE is written completely in [Python](https://www.edureka.co/blog/python-tutorial/) and it comes as a default implementation along with Python. Its name is presumed to be in honor of Eric Idle who is one of the founding members of Monty Python. This IDE is considered to be extremely suitable for the education industry due to its simplicity.

IDLE also provides some remarkable features such as:

* Availability of python shell with syntax highlighting
* A multi-window text editor
* Program animation or stepping (refers to executing one line of code at a time)
* Breakpoints are available to ease debugging
* Call stack is clearly visible

## 3.1 Python Basic Syntax

There is no use of curly braces or semicolon in Python programming language. It is English-like language. But Python uses the indentation to define a block of code. Indentation is nothing but adding whitespace before the statement when it is needed. **For example -**

def func():

       statement 1

       statement 2

       …………………

       …………………

        statement N

## Python Popular Frameworks and Libraries

* **Web development (Server-side) -** Django Flask, Pyramid, CherryPy
* **GUIs based applications -** Tk, PyGTK, PyQt, PyJs, etc.
* **Machine Learning -** TensorFlow, PyTorch, **Scikit-learn**, Matplotlib, Scipy, etc.
* **Mathematics -** Numpy, Pandas, etc.

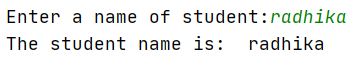
## Taking Input to the User

Python provides the **input()** function which is used to take input from the user.

**Example –**

name = input(**"Enter a name of student:"**)   
print(**"The student name is: "**, name)

**Output:**



Python Operators

Operators are the symbols which perform various operations on Python objects. Python operators are the most essential to work with the Python data types. In addition, Python also provides identify membership and bitwise operators.

Python Conditional Statements

Conditional statements help us to execute a particular block for a particular condition. In this tutorial, we will learn how to use the conditional expression to execute a different block of statements. Python provides if and else keywords to set up logical conditions. The elif keyword is also used as conditional statement.

## Python Loops

Sometimes we may need to alter the flow of the program. The execution of a specific code may need to be repeated several numbers of times. For this purpose, the programming languages provide various types of loops capable of repeating some specific code several times.

## Python Data Structures

Data structures are referred which can hold some data together or we say that they are used to store the data in organized way. Python provides built-in data structures such as **list, tuple, dictionary, and set**. We can perform complex tasks using data structures.

### Python List

Python list holds the ordered collection of items. We can store a sequence of items in a list. Python list is mutable which means it can be modified after its creation. The items of lists are enclosed within the square bracket [] and separated by the comma. Let's see the example of list.

If we try to print the type of L1, L2, and L3 using **type()** function then it will come out to be a list.

L1 = [**"John"**, 102, **"USA"**]  
L2 = [1, 2, 3, 4, 5, 6]  
print(type(L1))  
print(type(L2))

**Output:**



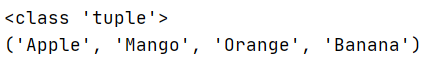
### Python Tuple

Python Tuple is used to store the sequence of immutable Python objects. The tuple is similar to lists since the value of the items stored in the list can be changed, whereas the tuple is immutable, and the value of the items stored in the tuple cannot be changed.

**Example -**

tup = (**"Apple"**, **"Mango"** , **"Orange"** , **"Banana"**)  
print(type(tup))  
print(tup)

**Output:**



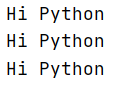
### Python String

Python string is a sequence of characters. It is a collection of the characters surrounded by single quotes, double quotes, or triple quotes. It can also define as collection of the Unicode characters. We can create a string as follows.

**Example –**

*# Creating string using double quotes*str1 = **"Hi Python"**print(str1)  
*# Creating string using single quotes*str1 = **'Hi Python'**print(str1)  
*# Creating string using triple quotes*str1 = **'''Hi Python'''**print(str1)

**Output:**



Python doesn't support the character data-type. A single character written as 'p' is treated as a string of length 1.

### Dictionaries

Python Dictionary is a most efficient data structure and used to store the large amount of data. It stores the data in the key-value pair format. Each value is stored corresponding to its key.

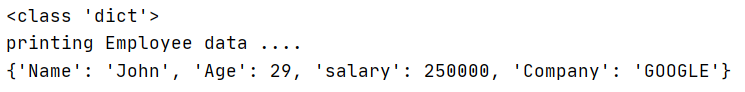
Keys must be a unique and value can be any type such as integer, list, tuple, etc.

It is a mutable type; we can reassign after its creation. Below is the example of creating dictionary in Python.

**Example –**

employee = {**"Name"**: **"John"**, **"Age"**: 29, **"salary"**:250000,**"Company"**:**"GOOGLE"**}  
print(type(employee))  
print(**"printing Employee data .... "**)  
print(employee)

**Output:**



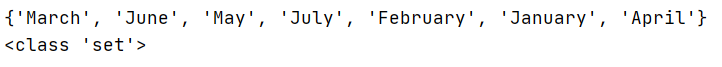
### Python Sets

A Python set is a collection of unordered elements. Each element in set must be unique and immutable. Sets are mutable which means we can modify anytime throughout the program.

**Example –**

*# Creating Set*Month = {**"January"**, **"February"**, **"March"**, **"April"**, **"May"**, **"June"**, **"July"**}  
print(Month)  
print(type(Month))

**Output:**



## Python File I/O

Files are used to store data in a computer disk. We can open a file using Python script and perform various operations such as writing, reading, and appending. There are various ways of opening a file. We are explained with the relevant example.

Python Modules

Python modules are the program files that contain a Python code or functions. There are two types of module in the Python - User-define modules and built-in modules. A module that the user defines, or we can say that our Python code saved with **.py** extension, is treated as a user-define module.

Built-in modules are predefined modules of Python. To use the functionality of the modules, we need to import them into our current working program.

## Python Exceptions

An exception can be defined as an unusual condition in a program resulting in the interruption in the flow of the program.

Whenever an exception occurs, the program stops the execution, and thus the further code is not executed. Therefore, an exception is the run-time errors that are unable to handle to Python script. An exception is a Python object that represents an error.

## Python Oops Concepts

Everything in Python is treated as an object including integer values, floats, functions, classes, and none. Apart from that, Python supports all oriented concepts. Below is the brief introduction of oops concepts of Python.

* **Classes and Objects -** Python classes are the blueprint of the object. An object is a collection of data and method that act on the data.
* **Inheritance -** An inheritance is a technique where one class inherits the properties of other classes.
* **Constructor -** Python provides a special method **\_\_init\_\_()** which is known as a constructor. This method is automatically called when an object is instantiated.
* **Data Member -** A variable that holds data associated with a class and its objects.

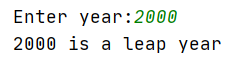
**4. Assignments and project**

# Python assignments

4.1 Leap year

*# Python program to check if year is a leap year or not*year = int(input(**"Enter year:"**))  
  
*# To get year (integer input) from the user  
# year = int(input("Enter a year: "))*if (year % 4) == 0:  
 if (year % 100) == 0:  
 if (year % 400) == 0:  
 print(**"{0} is a leap year"**.format(year))  
 else:  
 print(**"{0} is not a leap year"**.format(year))  
 else:  
 print(**"{0} is a leap year"**.format(year))  
else:  
 print(**"{0} is not a leap year"**.format(year))

Output



4.2 Try and except

*# Python code to illustrate  
# working of try()*def divide(x, y):  
 try:  
 *# Floor Division : Gives only Fractional Part as Answer* result = x // y  
 print(**"Yeah ! Your answer is :"**, result)  
 except ZeroDivisionError:  
 print(**"Sorry ! You are dividing by zero "**)  
  
 *# Look at parameters and note the working of Program*divide(3, 0)

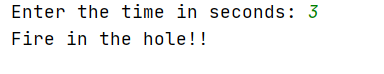
Output-



4.3 Timer

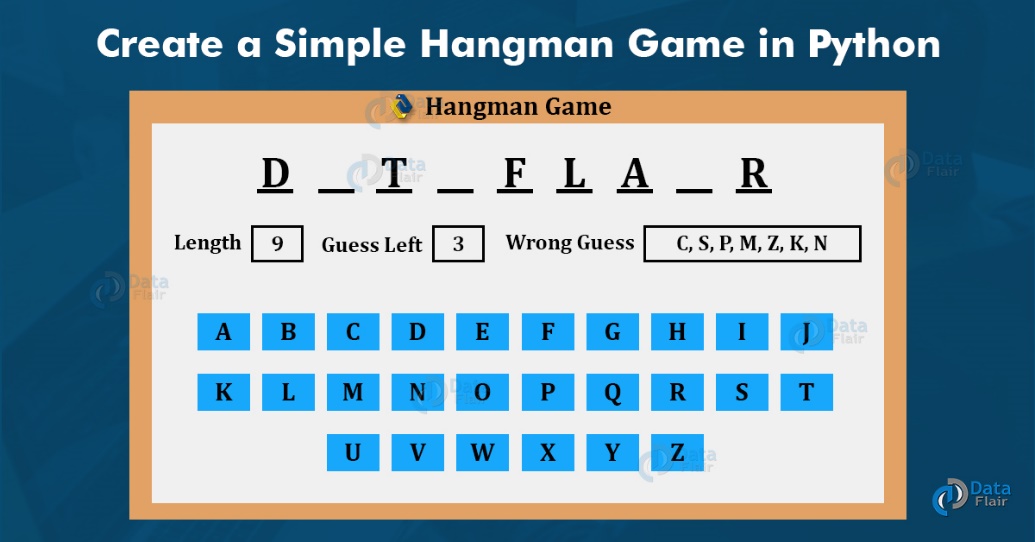
*# import the time module*import time  
  
  
*# define the countdown func.*def countdown(t):  
 while t:  
 mins, secs = divmod(t, 60)  
 timer = **'{:02d}:{:02d}'**.format(mins, secs)  
 print(timer, end=**"**\r**"**)  
 time.sleep(1)  
 t -= 1  
  
 print(**'Fire in the hole!!'**)  
  
  
*# input time in seconds*t = input(**"Enter the time in seconds: "**)  
  
*# function call*countdown(int(t))

Output -



# **4.4 Project**

# Hangman Game in Python

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2020/07/simple-hangman-game-python.jpg)

### About Hangman

Going back to our old school days, some of the pen-paper games were always a top for our leisure time. Hangman was one, other than some chit games, to guess words according to the guesses determined and as soon as they lost all their wrong guesses, they were hanged (not really, but on paper ). That is an old way, now to play Hangman. The new advancement in technologies allows us to play hangman using our own computer also without any other player. How? Let’s find out further

### Python Hangman Project

The objective of this project is to implement the hangman game using Python. It doesn’t require any specific modules other than random and time. Python loops and functions are enough to build this game here.

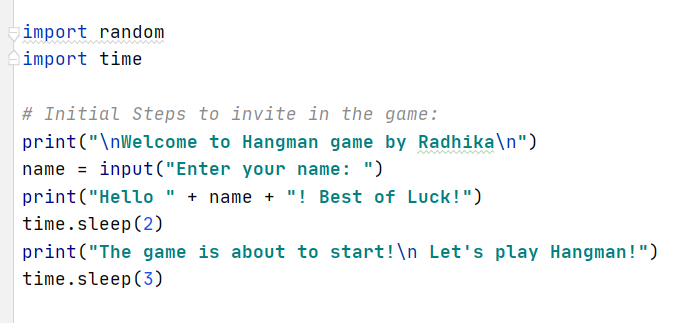
### Project Prerequisites

This project requires good knowledge of Python which includes defining functions and managing for/while loops. The functions that are used here contain arguments that are defined in a global scope which can be further used in other functions to improve game quality. It can also be used to provide different steps when required to execute upon conditions by the for and while loops.

### Project File Structure

So that is basically what is done in this Python project. Let’s start.

1. Importing the random and time libraries:



**Code Explanation:**

* **Import random:** This is used to randomly choose an item from a list [] or basically a sequence.
* **Import time:** This module is used to import the actual time from your pc to use in the program.
* **Time.sleep():** This is used to halt the execution of the program for a few seconds. It is a fun way to put the user of the game in short suspense.

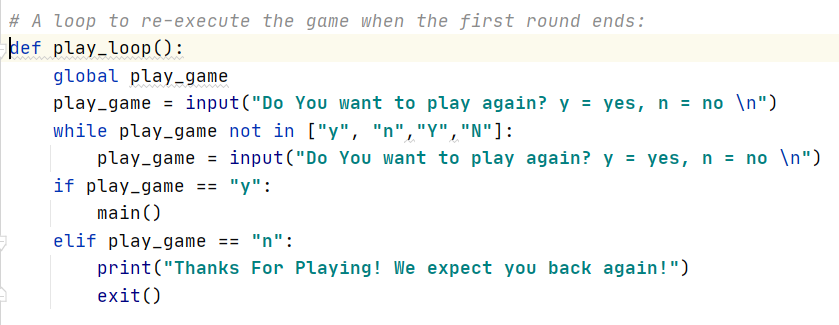
1. Define the main function:



**Code Explanation:**

* We define the main function that initializes the arguments: global count, global display, global word, global already\_guessed, global length and global play\_game. They can be used further in other functions too depending on how we want to call them.
* **Words\_to\_guess:** Contains all the Hangman words we want the user to guess in the game.
* **Word:** we use the random module in this variable to randomly choose the word from words\_to\_guess in the game.
* **Length:** len() helps us to get the length of the string.
* **Count:** is initialized to zero and would increment in the further code.
* **Display:** This draws a line for us according to the length of the word to guess.
* **Already\_guessed:** This would contain the string indices of the correctly guessed words.

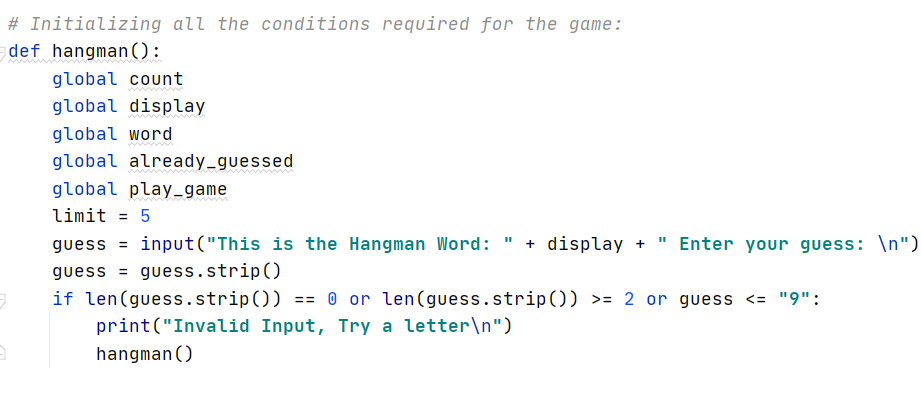
1. Develop a loop to execute the game again:



**Code Explanation:**

* **Play\_loop:** This function takes in the argument of play\_game.
* **Play\_game:** We use this argument to either continue the game after it is played once or end it according to what the user suggests.
* While loop is used to execute the play\_game argument. It takes the parameter, y=yes and n=no. If the user gives an input of something else other than y/n, it asks the question again for the appropriate answer. If the user inputs “y”, the game restarts, otherwise the game ends.

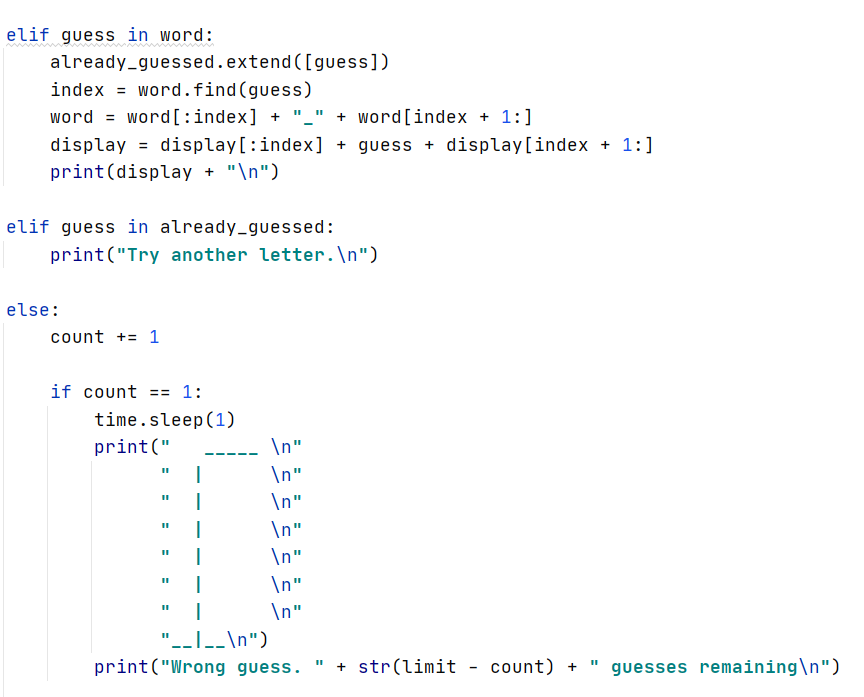
1. Initialize conditions for hangman game:

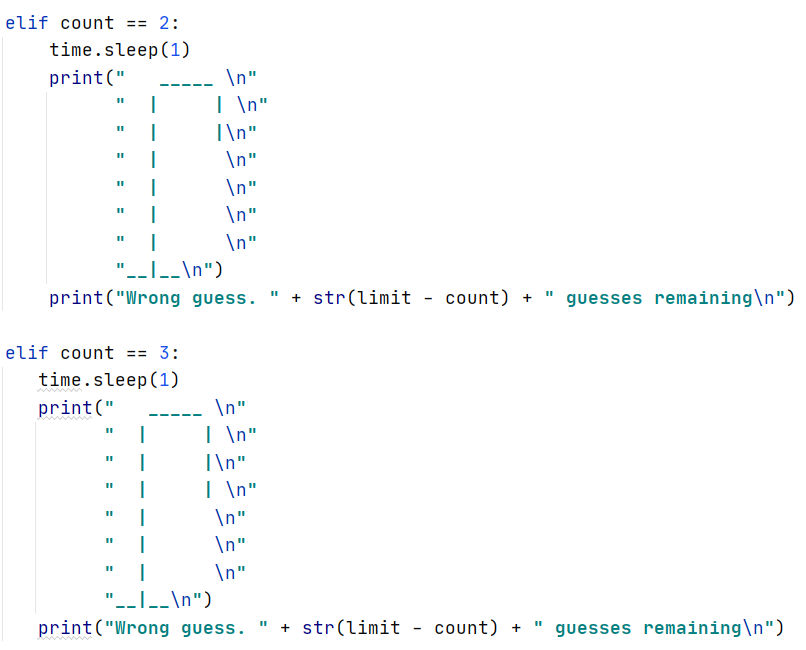


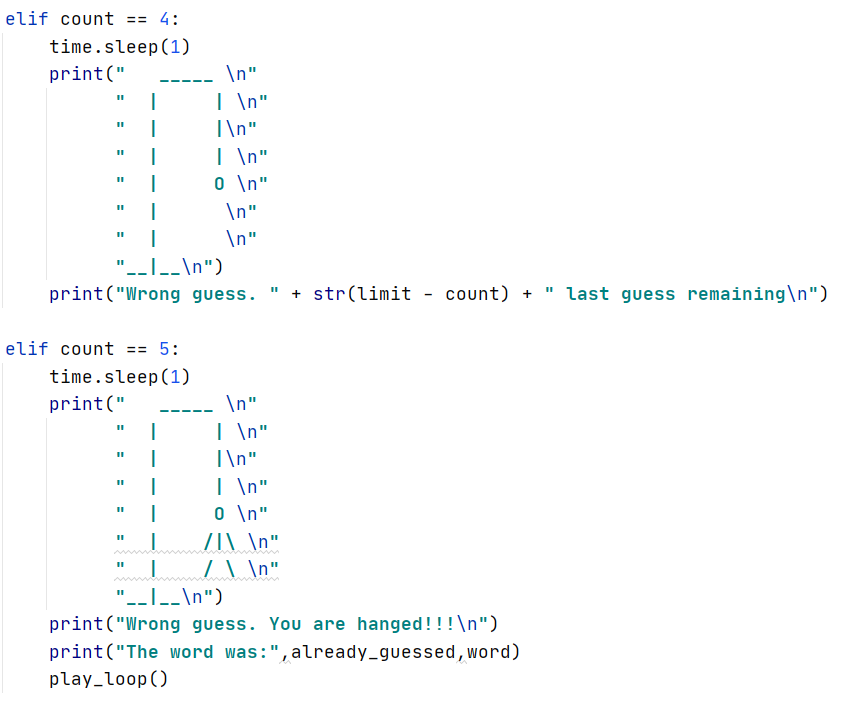
**Code Explanation:**

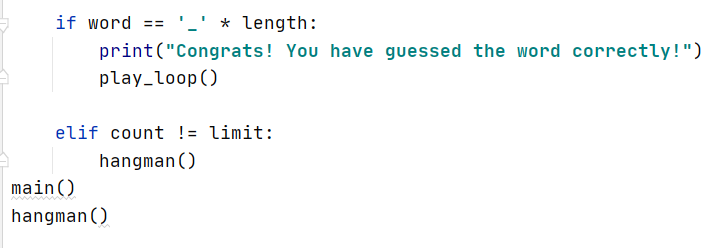
* We call all the arguments again under the **hangman() function**.
* **Limit:** It is the maximum guesses we provide to the user to guess a particular word.
* **Guess:** Takes the input from the user for the guessed letter. **Guess.strip()** removes the letter from the given word.
* If loop checks that if no input is given, or two letters are given at once, or a number is entered as an input, it tells the user about the invalid input and executes hangman again.

1. The rest of the whole hangman program combined together:





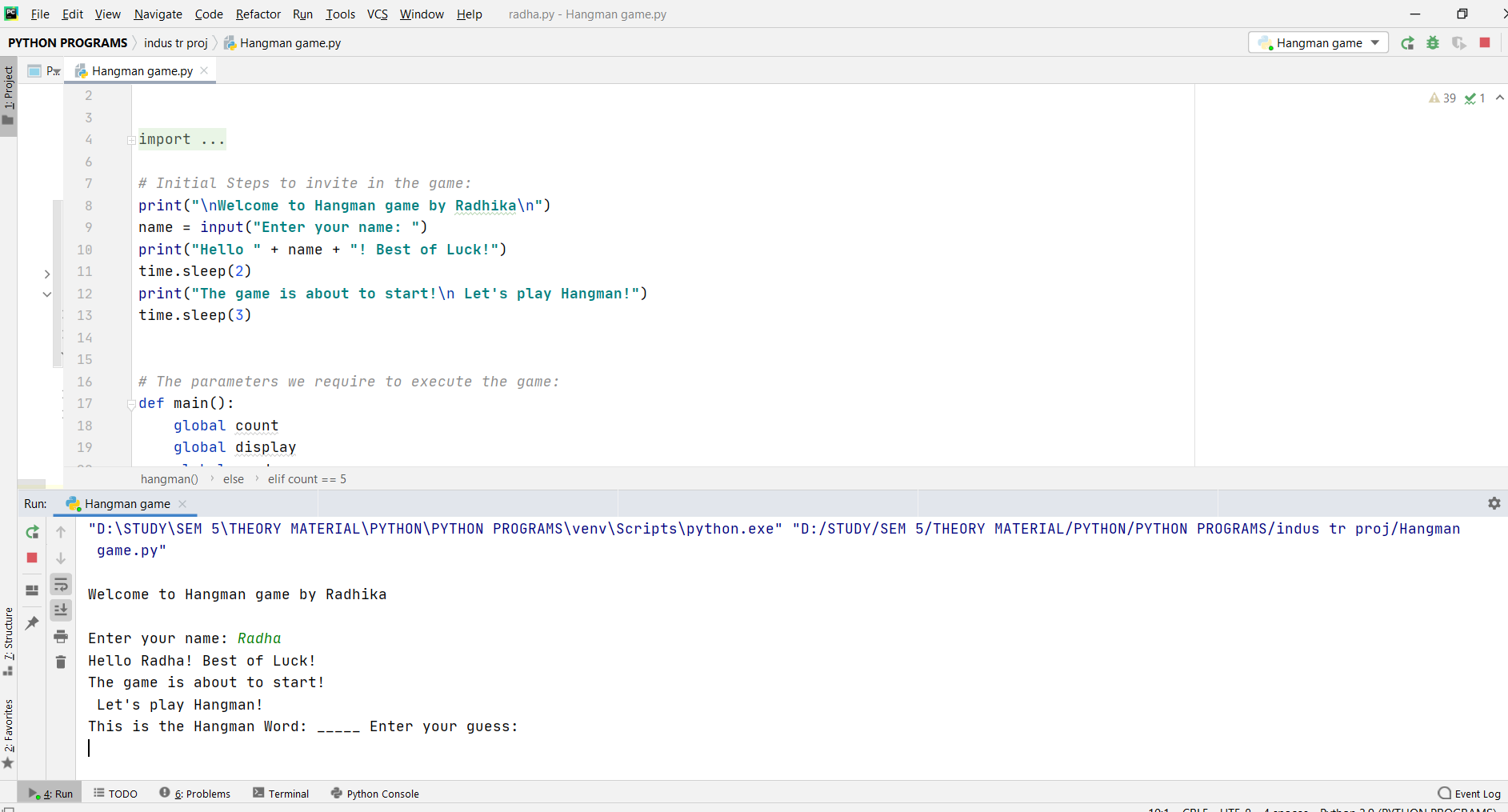




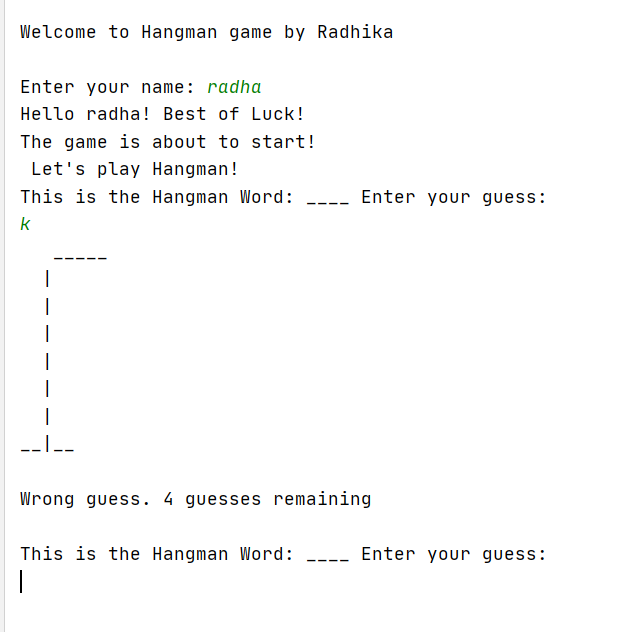
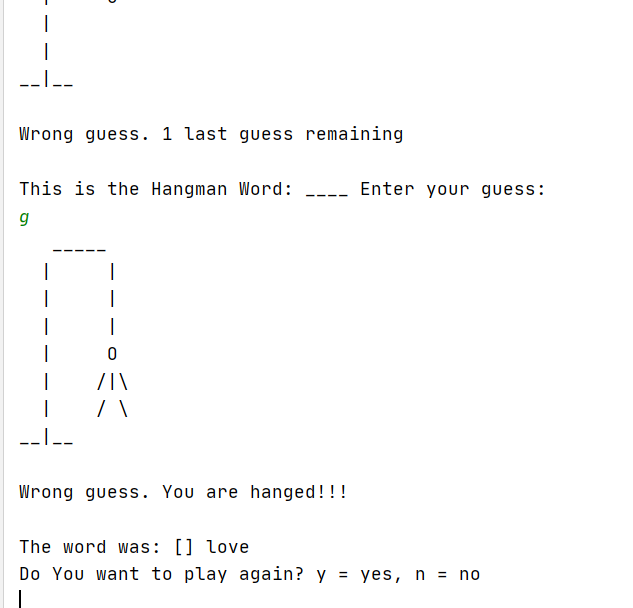
**Code Explanation:**

* If the letter is correctly guessed, index searches for that letter in the word.
* Display adds that letter in the given space according to its index or where it belongs in the given word.
* If we have already guessed the correct letter before and we guess it again, It tells the user to try again and does not lessen any chances.
* If the user guessed the wrong letter, the hangman starts to appear which also tells us how many guesses are left. Count was initialized to zero and so with every wrong guess its value increases with one.
* Limit is set to 5 and so (limit- count) is the guesses left for the user with every wrong input. If it reaches the limit, the game ends, showing the right guesses (if any) and the word that was supposed to be guessed.
* If the word is guessed correctly, matching the length of the display argument, the user has won the game.
* Play\_loop asks the user to play the game again or exit.
* Main() and hangman() would start again if the play\_loop executes to yes.

### Project output



Hangman Game various outputs:

With this project in Python, I have successfully developed the Hangman game. I have used the popular time and random modules to render the program. Executing different functions and using loops helped me with a better understanding of python basics.

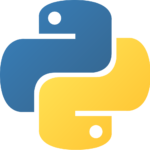
## 5.Conclusion Future Work

With the rising popularity and boon of [Python](https://www.edureka.co/python), it has become one of the most sought-after and widely used programming languages in the industry. Creating a euphoria among developers, one is bound to wonder what are the Python Developer skills in order to become a Python developer.

The skills I would like to work on in the future are mentioned below-

5.1 Expertise in Core Python

This forms the stepping stone for becoming a Python developer.



* Data Structures
* [OOPs concepts](https://www.edureka.co/blog/object-oriented-programming-python/)
* [Variables and data types](https://www.edureka.co/blog/variables-and-data-types-in-python/)
* [File handling concepts](https://www.edureka.co/blog/file-handling-in-python/)
* [Exception Handling](https://www.edureka.co/blog/exceptions-in-python/)
* Iterators

5.2 Good Grasp of Web Frameworks



Django is a high-level Python Web Framework that encourages a good, clean and pragmatic design and Flask is also widely used Python micro web framework. Sound knowledge of  Front-end technologies like [HTML](https://www.edureka.co/blog/what-is-html/), [CSS](https://www.edureka.co/blog/what-is-css/), and[JavaScript](https://www.edureka.co/blog/what-is-javascript/) is also expected.

5.3 Road to Data Science



[Data Science](https://www.edureka.co/blog/what-is-data-science/) is an ocean of opportunities. Once you deep dive in the same then there are certain prerequisites you should know starting from your high-school mathematics consisting of Probability, Statistics, etc. Other most common aspects of the same are.

* Visualization of Data
* Analysis of Data
* Data Wrangling and cleaning up of Data
* Use of Python packages like([NumPy](https://www.edureka.co/blog/python-numpy-tutorial/), [MatPlotlib](https://www.edureka.co/blog/python-matplotlib-tutorial/" \t "_blank), [Scikit learn](https://www.edureka.co/blog/scikit-learn-machine-learning/), etc) for Data Science.
* Good Knowledge of [SQL](https://www.edureka.co/blog/mysql-tutorial/).

5.4 Machine Learning and AI



When we talk about[Machine Learning](https://www.edureka.co/blog/machine-learning-tutorial/) and [Artificial Intelligence](https://www.edureka.co/blog/artificial-intelligence-with-python/) fields who happen to fall under Data Science. It requires a good understanding of how to acquire data from datasets, analyze data, visualize data, produce insights from data, learn about [Neural Networks](https://www.edureka.co/blog/neural-network-tutorial/), etc.

* 1. Analytical Skills

To become a good python developer or be it any programming language expert. One should have good analyzing skills in terms of Python it includes. Good understanding of algorithms so that you write clean and not redundant code, able to write optimized algorithms, Visualise datasets in a better way w.r.t to Data Science, Build good websites w.r.t web development.

5.6 Design Skills



What I mean by good design skills is related to the fact that I should be able to design scalable products also, implement servers in such a way that they are highly available. One should also keep in mind the frameworks of python like [Django](https://www.edureka.co/blog/django-tutorial/) or [Flask](https://www.youtube.com/watch?v=lj4I_CvBnt0) while designing a website as python can work in both client and [server-side programming](https://www.edureka.co/blog/socket-programming-python/#socketmodule).

5.7 Communication Skills



One of the most important aspects of any profession largely depends on having really good communication skills. If you are able to contribute within the team, do peer code review, communicate in an efficient manner then half of your job is done there itself. Even in a core technical role you should specifically communicate with your teammates and contribute to resolve issues or help others.