

PROFESSIONAL TRAINING REPORT

at

Sathyabama Institute of Science and Technology

(DEEMED TO BE UNIVERSITY)

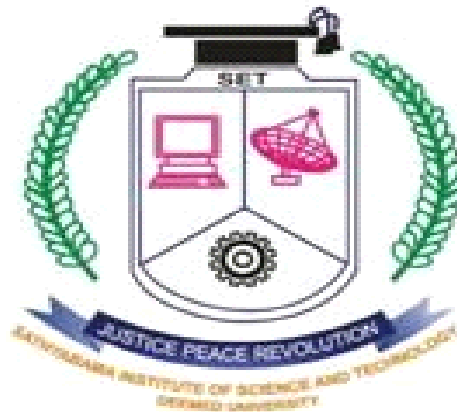
Submitted in partial fulfilment of the requirements for the award of Bachelor of
Technology Degree in

INFORMATION TECHNOLOGY

By

BATHULA SOWMYA REDDY

(REG NO:37120013)



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI – 600119, TAMILNADU.

AUGUST 2019



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" Grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119.

Phone: 044 - 2450 3150 / 51 / 52 / 54 / 55 Fax: 044 - 2450 2344

www.sathyabama.ac.in



SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work **BATHULA SOWMYA REDDY (Reg.No.37120013)** who underwent the professional training in "INTERACTIVE DICTIONARY" under our supervision from July 2020 to August 2020

Internal Guide

Dr.S. REVATHY .M.E.,Ph.D.,

Head of the Department

Dr.R.SUBHASHINI. M.E.Ph.d.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **BATHULA SOWMYA REDDY (Reg.No:37120013)** hereby declare that the Professional Training Report on “**INTERACTIVE DICTIONARY**” done by me under The Guidance of guide **Dr. S. REVATHY**, Dept. of IT at Sathyabama Institute of Science and Technology is submitted in partial fulfilment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE:
PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

-

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D, Dean, School of Computing , Dr.SUBHASHINI M.E., Ph.D. ,** Head of the Department of Information and Technology for providing me necessary support and details at the right time during the progressive reviews.

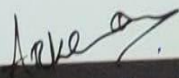
I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. Revathy M.E., Ph.D.,**for his valuable guidance, suggestions and constant encouragement paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Information Technology who were helpful in many ways for the completion of the project.

CERTIFICATE OF MERIT

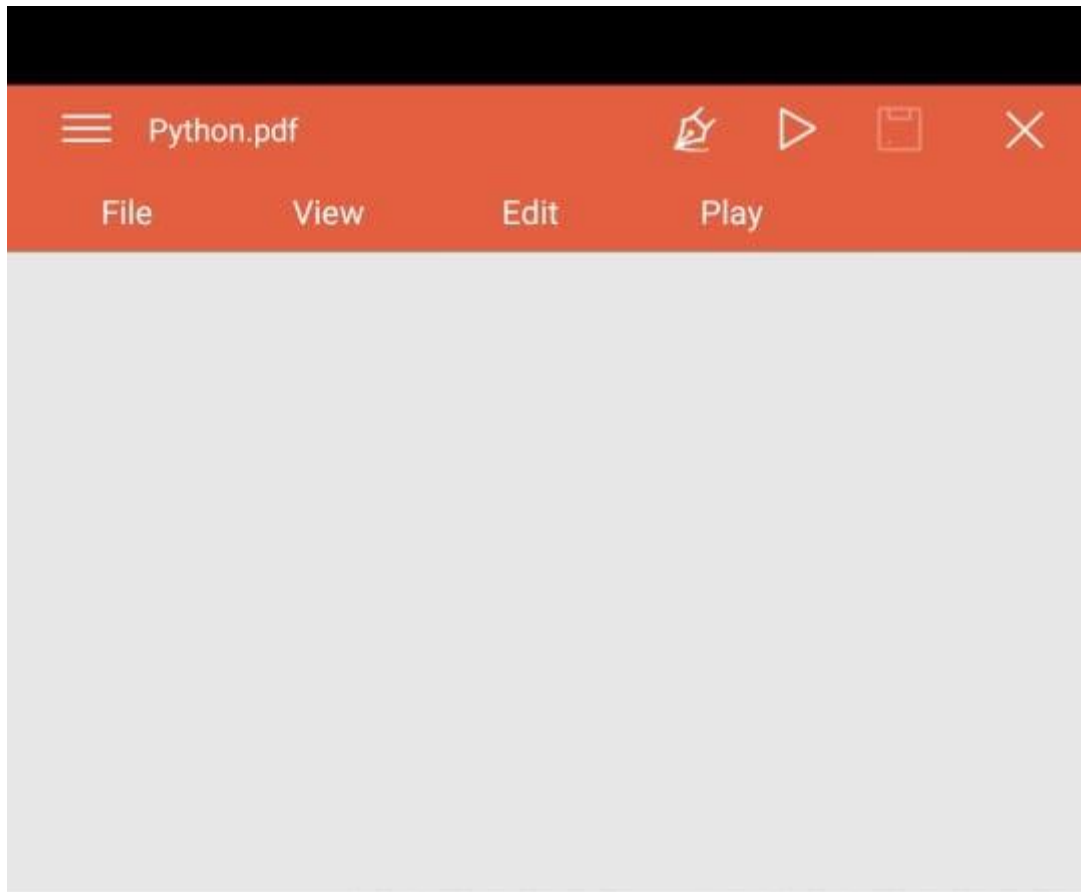
This is to certify that B. SOWMYA REDDY (Reg No. 175IT001) has
successfully completed the internship in JAVA Application Development in our
concern from 03-06-19 to 12-06-19.

During the internship period, the performance of the intern was found to be Good.


Program Coordinator



HR Head



ABSTRACT

INTERACTIVE DICTIONARY main purpose of the project is how to get the meaning of the words that are present in the java script object notation file. If any wrong word is entered then it checks for their closely related words for the entered word and display. If that word one is looking for then choose the index so that it prints the meaning of the word. This becomes very handy if one wants to find the meaning of the particular word and can be able to find the meaning very quickly by entering the word. Python also provides very handy features for developing the code by importing the code into predefined modules. Although python is the one of the easiest and flexible languages for the users to learn because it provides easy syntax and the rules.

TABLE OF CONTENT

CH NO	TITLE NAME	PG NO
	ABSTRACT	I
	LIST OF FIGURES	II
	LIST OF ABBRIVATIONS	III
1	INTRODUCTION	1
	1.1 HISTORY OF PYTHON	1
	1.2 FEATURES OF PYTHON	1
	1.3 OUTLINE OF THE PROJECT	2
	1.4 LITERATURE REVIEW	2
2	AIM AND SCOPE OF PRESENT OVERVIEW	
	2.1 AIM OF THE PROJECT	3
	2.2 OVERVIEW	3
	2.2.1 DICTIONARIES	3
	2.2.2 JSON	4
	2.3 SCOPE OF THE PROJECT	4
3	SYSTEM IMPLEMENTATION	
	3.1 GENERAL	5
	3.2 OVERVIEW OF THE PLATFORM	5
	3.2.1 CONDITIONAL STATEMENTS	5
	3.2.2 LOOPING STATEMENTS	8
	3.2.3 FUNCTIONS	9
	3.2.4 LISTS	11

	3.2.5 STRINGS	11
	3.2.6 DIFFLIB MODULE	12
4	RESULTS AND DISCUSSIONS	
	4.1 SCREENSHOTS	14
5	CONCLUSION AND FUTURE WORK	17
	REFERENCE	18
	APPENDIX	19
	SOURCE CODE	19

LISTS OF FIGURES

FIG NO	FIGURE NAME	PAGE NO
2.1	ILLUSTRATION OF DICTIONARIES	4
3.1	IF CONDITION FLOWCHART	5
3.2	ELIF CONDITION STATEMENT	6
3.3	ELSE CONDITION FLOWCHART	7
4.1	SAMPLE OUTPUT-1	14
4.2	SAMPLE OUTPUT-2	15
4.3	SAMPLE OUTPUT-3	15
4.4	SAMPLE OUTPUT-4	16

LIST OF ABBREVIATIONS

CWI	-	CENTRUM WISKUNDE AND INFORMATICA
JSON	-	JAVASCRIPT OBJECT NOTATION
GUI	-	GRAPHICAL USER INTERFACE

CHAPTER 1

INTRODUCTION

1.1 HISTORY OF PYTHON

Python was conceived in the late 1980s, and its implementation began in December 1989 by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL) capable of exception handling and interfacing with the Amoeba operating system. Python 3.0 (initially called Python 3000 or py3k) was released on 3 December 2008 after a long testing period. It is a major revision of the language that is not completely backward-compatible with previous versions. However, many of its major features have been backported to the Python 2.6.x and 2.7.x version series, and releases of Python 3 include the 2to3 utility, which automates the translation of Python 2 code to Python3

1.2 FEATURES OF PYTHON

Python is a multi- paradigm programming language. Object- oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming. Many other paradigms supported via Python use dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has `filter()`, `map()`, and `reduce()` functions; list comprehensions, dictionaries, and sets; and generator expressions.

The standard library has two modules (`itertools` and `functools`) that implement functional tools borrowed from Haskell and Standard ML.

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

1.3 OUTLINE OF THE PROJECT

The main goal of the project is to get the meanings of the words from the given "JSON" file using the dictionary's concept by importing the modules named json and get_close_matches. It allows the user to get the synonyms of the particular word. The functioning of the project is divided into three main modules:-

1. Frontend
2. Backend
3. Connecting backend and frontend

1.4 LITERATURE REVIEW

In the project the main concepts used are Python dictionaries and the jsonfile. JSON, as a format is very similar to python dictionaries, so similar in fact there's a very simple way to work with it. If you're already working with dictionaries this will feel like an easy transition. There are, however, a few things you need to keep in mind. Make sure that our JSON is correctly formatted. However dictionaries contains the key,value pairs. The Json module provides an easy way to encode and decode data in JSON. The two main functions are json.dumps() and json.loads(), mirroring the interface used in other serialization libraries, such a pickle.

CHAPTER 2

AIM AND SCOPE OF PRESENT OVERVIEW

2.1 AIM OF THE PROJECT

This system provides the meanings or synonyms of a particular word from the provided json file. If this program is run in the python module it gets the meaning of the entered word. Suppose a wrong word is entered it shows the closely related words that are present, So that it can be able to choose the word accordingly by selecting the index of the particular word. If any word has more than one meanings then it displays the meanings one of the another.

2.2 OVERVIEW

The main module of the project is the json module and difflib in this by importing `get_close_matches` which identifies the closest possible words from the json file when a wrong word is entered. The concepts used in this are dictionaries and If loops and Functions. The database used in this is JSON file which stores the key,value pairs of the dictionaries. Using this JSON file, all the meanings that are available can be viewed and could be able to add more meanings and could also be able to modify

.2.2.1 DICTIONARIES

Dictionaries are mutable unordered collections of key-value pairs. Keys with in the dictionary must be unique and must be hashable. That includes types like numbers, strings and tuples. Lists and dicts cannot be used as keys since they are mutable. Dictionaries in other languages are also called as the associative arrays. Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces. An empty dictionary without any items is written with just two curly braces, like this: `{}`. The values of a dictionary can be of any type, but keys must be of an immutable data type such as strings, numbers or tuples.

Example:

```
Dict={'name': 'zara', 'Age': '7', 'class': 'Second'}
```

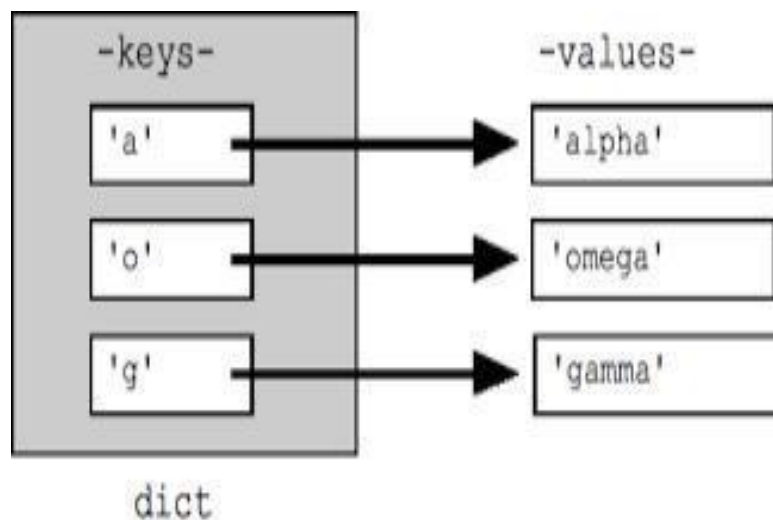



Fig. 2.1 Illustration of dictionaries

The above figure describes the key, value pairs in dictionaries.

2.2.2 JSON

JavaScript Object Notation is an easy, flexible to read, flexible text based format that can be used to store and communicate information to other products. It is mainly based on key: value pairs and is web and .NET friendly. There are many libraries and products that support JSON.

- JSON can store Lists, bools, numbers, tuples and dictionaries. But to be saved into a file, all these structures must be reduced to strings. It is the string version that can be read or written to a file. Python has a JSON module that will help converting the data structures to JSON module function to strings. Use the import the JSON
- The JSON module is mainly used to convert the python dictionary above into a JSON string that can be written into a file. `json.dumps(datastore)`
- The json module can also take a string and convert it back to a dictionary structure. `json.load(json_string)`
- While the JSON module will convert strings to Python data types, normally the JSON functions are used to read and write directly from JSONfiles.

2.3 SCOPE OF THE PROJECT

The present dictionary interface project can be developed as a web application by further adding GUI to it. If a GUI is added it can be developed as a web page by creating the boxes to it i.e, it can be developed as if the word is entered it will display the further meanings of that word in the boxes.

CHAPTER 3

SYSTEM IMPLEMENTATION

3.1 General

In this project the code is written using python3 and the concepts the main concepts that are used are dictionaries,lists,functions,conditional statements and modules have to be imported like json and the other module difflib. Difflib is used to the get_close_matches for identifying the closest matching words in the json file. Run this in the python3 IDLE by clicking the run button in the file where it returns the source code. The main thing was that the both JSON file and the python source code file have to be placed in the same folder in our pc.

3.2 OVERVIEW OF THE PLATFORM

3.2.1 CONDITIONAL STATEMENTS

A statement that controls the flow of execution depending on some condition.

TYPES:

1. If Statement
2. Elif Statement
3. Else Statement

1. IF STATEMENT

In order to write useful programs, its always needed the ability to check conditions and change the behavior of the program accordingly. Here comes the use of if statement

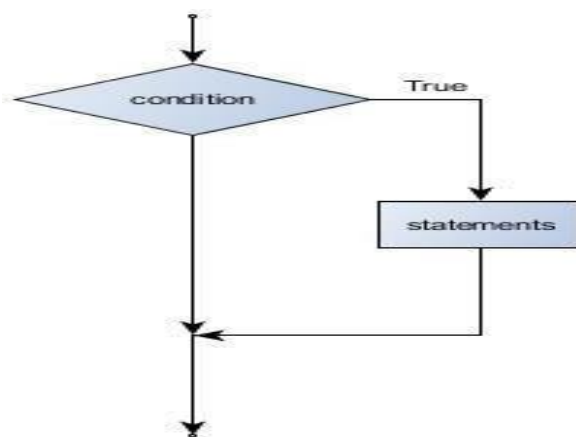


Fig. 3.1 If condition flowchart

This above flow chart describes a the execution of an if statement

Syntax

```
If  BOOLEAN EXPRESSION:
    STATEMENTS
```

A few important things to note about if statements

- The colon (:) is significant and required. It separates the header of the compound statement from the body.
- The line after the colon must be indented. It is standard in Python to use four spaces for indenting.
- All lines indented the same amount after the colon will be executed whenever the `BOOLEAN_EXPRESSION` is true.

2. ELIF STATEMENT

Sometimes there are more than two possibilities and more than two branches are needed. Here comes the use of elif statement.

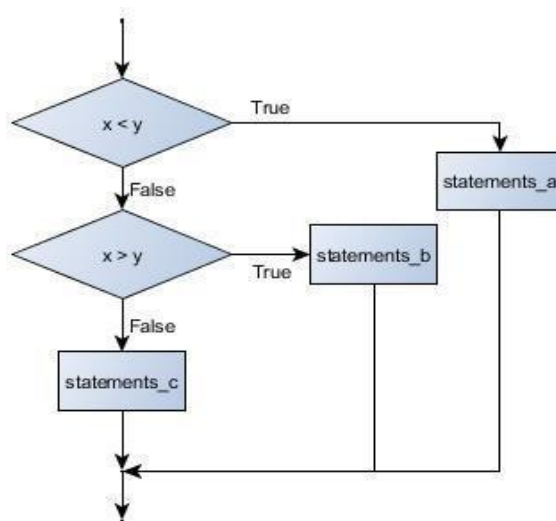


Fig. 3.2 Elif condition flowchart

This above flow chart describes a the execution of an elif statement

Syntax

```
if  BOOLEAN EXPRESSION:
    STATEMENTS
elif BOOLEAN EXPRESSION:
    STATEMENTS
else:
    STATEMENT
```

□ elif is an abbreviation of else if. Again, exactly one branch will be executed. There is no limit to the number of elif statements but only a single (and optional) final else statement is allowed and it must be the last branch in the statement.

3. ELSE STATEMENT

It is frequently the case that you want one thing to happen when a condition is true, and something else to happen when it is false. For that the if else statement is needed.

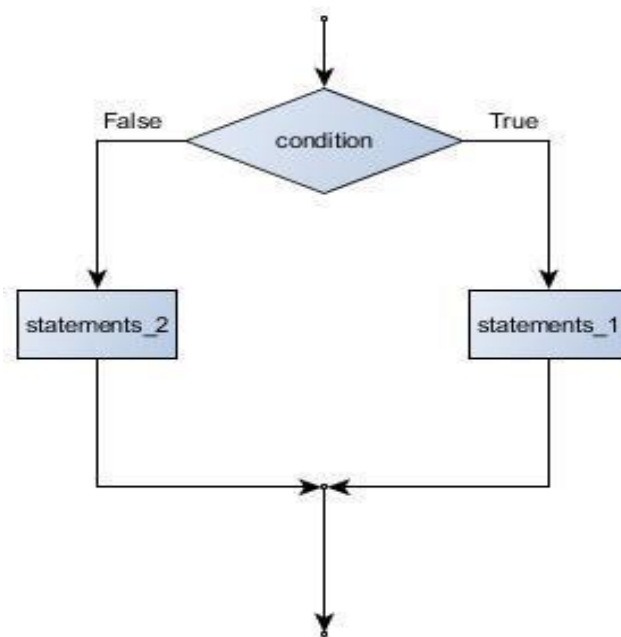


Fig. 3.3 Else condition flowchart

This above flow chart describes the execution of an else statement

Syntax

```
if      BOOLEAN
    EXPRESSION: STATEMENTS
else:
    STATEMENTS
```

- In this at first 'if' statement is executed if the 'if' statement is true then it is executed and else statement is skipped. If it is false then else statement is executed.
- In the conditional statement only one else statement will exist. Hence more than one if and elif statement are needed.

- Elseif statement can also be used in the place of the elif statement in some of the cases.

3.2.2 LOOPING STATEMENTS

Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body. Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.

Types:

1. For loop
2. While loop

1. FOR LOOP

Python for loop is used to iterate the elements of a collection in the order that they appear. This collection can be a sequence (list or string).

Syntax

for<variable> in <sequence>:

STATEMENTS:

Way of execution:

- Firstly, the first value will be assigned in the variable
- Secondly all the statements in the body of the loop are executed with the same value.
- Thirdly, once step second is completed then variable is assigned then next value in the sequence and step second is repeated.
- Finally, it continues till all the values in the sequence are assigned in the variable and processed.

2. WHILE LOOP

In Python, while loop is used to execute number of statements or body till the specified condition is true. Once the condition is false, the control will come out of the loop.

Syntax

while<expression>:

STATEMENTS

Rules to be followed

- Firstly, the value in the variable is initialized.
- Secondly, the condition/expression in the while is evaluated. Consequently, if the condition is true, the control enters in the body and executes all the statements. If the condition/expression passed results in false then the control exists the body and straight away control goes to the next instruction after a while.
- Thirdly, in case the condition was true having completed all the statements, the variable is incremented or decremented. Having changed the value of variable step second is followed. This process continues till the expression/condition becomes false.
- Finally Rest of code after the body is executed.

3.2.3 FUNCTIONS

A Function is a self-block of code which is used to organize the functional code. Function can be called as a section of a program that is written once and can be executed whenever required in the program, thus making re-usability. Function is a subprogram that works on data and produces some output.

Types of Functions

a) Built-in Functions

Functions that are predefined and organized into a library. Many predefined functions have been used in Python.

b) User-Defined

Functions that are created by the programmer to meet the requirements.

DEFINING A FUNCTION

A Function defined in Python should follow the following format

- Keyword def is used to start and declare a function. Def specifies the starting of function block.
- def is followed by function-name followed by parenthesis.
- Parameters are passed inside the parenthesis. At the end a colon is marked.
- Python code requires indentation (space) of code to keep it associate to the declared block.

- The first statement of the function is optional. It is? Documentation string? of function.
- Following is the statement to be executed.

Default Function Syntax

```
def<function name>():  
STATEMENTS
```

Return Statement

Return[expression] is used to return response to the caller function. The return keyword can be used with the expression. Send back the control to the caller with the expression. In case no expression is given after return it will return None. In other words return statement is used to exit the function definition.

Syntax

```
def<function name>(parameters):  
return [expression_list].
```

PRE-DEFINED FUNCTIONS

The functions that are already defined in the compiler are called predefined functions.

1. PRINT FUNCTION

The print function in Python is a function that outputs to your console window whatever you say you want to print out. At first blush, it might appear that the print function is rather useless for programming, but it is actually one of the most widely used functions in all of python.

Syntax

```
print(<statement, data types>)
```

2. INPUT FUNCTION

Input can come in various ways, for example from a database, another, mouse clicks and movements or from the internet. Yet, in most cases the input stems from the keyboard. For this purpose, Python provides the function input(). input has an optional parameter, which is the prompt string.

Syntax

```
a=input(<Statements>)
```

3.2.4 LISTS

Lists are one of the most powerful tools in python. They are just like the arrays declared in other languages. But the most powerful thing is that list need not be always homogeneous. A single list can contain strings, integers, as well as objects. Lists can also be used for implementing stacks and queues. Lists are mutable, i.e., they can be altered once declared.

Syntax

`L=[<integer,float,string etc..>]`

- The index operator[] could be used to access an item in a list. Index starts from 0. So, a list having 5 elements will have index from 0 to 4.
- Trying to access an element other than this will raise an index error. The index must be an integer. The float or other types can't be used, this will result into Type error.

Build-in-List-Functions

- `Append()`: Add an element to the end of list
- `Extend()`: Add all elements of a list to the another list
- `Insert()`: Insert an item at the defined index
- `Remove()`: Removes an item from the list
- `Pop()`: Removes and returns an element at the given index
- `Clear()`: Removes all items from list
- `Index()`: Returns the index of the first matched item
- `Count()`: Returns the count of number of items passed as an argument
- `Sort()`: Sort items in a list in ascending order
- `Reverse()`: Reverse the order of items in the list
- `Copy()`: Returns a shallow copy of list.
- A list can even have another list as an item. This is called nested list.

Example:

`A=[1234,['chennai',33],'mumbai',[1,2,3]]`

- Python also allows negative indexing for its sequences. The index of -1 refers to the last item, -2 to the second last item and soon.

3.2.5 Strings

Strings in python are arrays of bytes representing unicode characters. However, python does not have a character data type, a single character is simply

string with a length of 1. Square brackets can be used to access elements of the string. You can update an existing string by reassigning a variable to another string. The new value can be related to its previous value or to a completely different string altogether.

Build-in-String-Functions

- ❑ `letters()`: concatenation of the strings lowercase and uppercase
- ❑ `lowercase()`: Returns the string in lowercase
- ❑ `uppercase()`: Returns the string in uppercase
- ❑ `len()`: Returns the length of the whole string
- ❑ `strip()`: Method removes any white spaces at beginning or end
- ❑ `replace()`: This method replaces the string with other string
- ❑ `split()`: This method divides the string into sub strings if it finds the instance of separators.
- ❑ `max()`: Returns the highest alphabetical character in string
- ❑ `min()`: Returns the lowest alphabetical character in the string
- ❑ `count()`: Returns the number of occurrences of substring in a string
- ❑ `Isalpha()`: Returns "True" if all characters in the string are alphabets, Otherwise, It returns "False".
- ❑ `Isdecimal()`: Returns if True of all characters in a string are decimal.

3.2.6 DiffLibModule

This module provides classes and functions for comparing sequences. It can be used for example, for comparing files, and can produce difference information in various formats, including HTML and context and unified diffs. For comparing directories and files.

This module has to be imported to use this

`*Import`

`difflib.get_close_matches`

`s()`:

This consists of the arguments are words, possibilities, n, cutoff ratio

Syntax

`get_close_matches(words, possibilities, n=3, cutoff=0.6)`

The default values of n and cutoff is '3' and '0.6' respectively

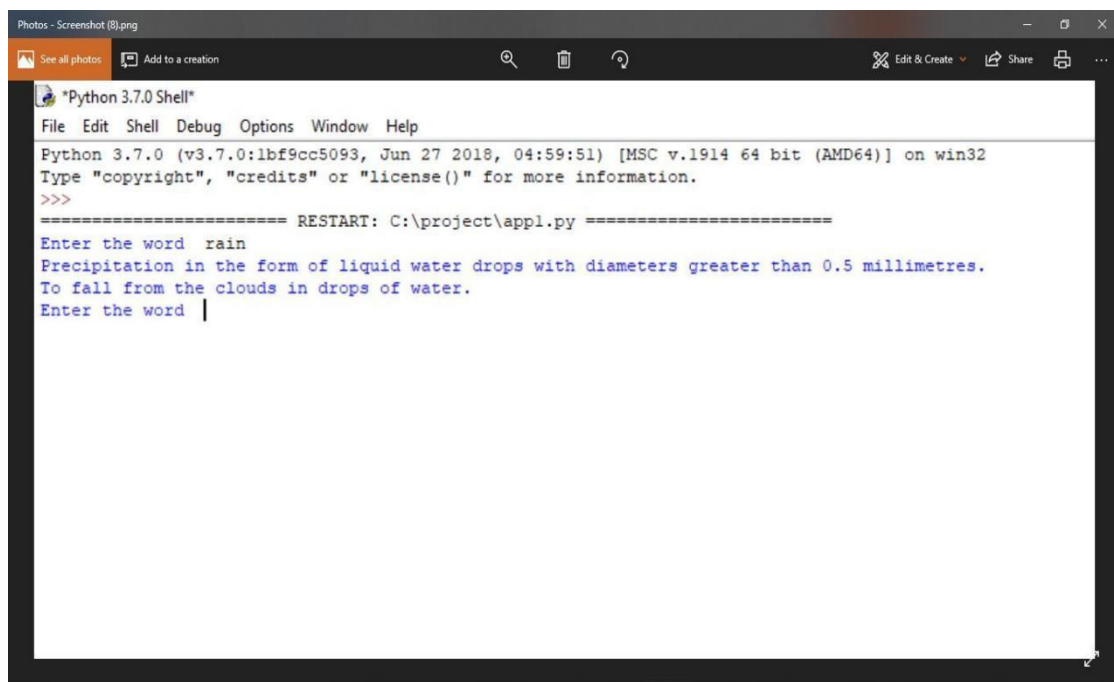
- This will return a list of best good enough matches. *Word* is a sequence for which close matches are desired(String) and *possibilities* is list of sequences against which to match *word*
- Optional argument *n* (default 3) is the maximum number of close matches to return; *n* must be greater than 0.
- Optional argument *cutoff* (default 0.6) is a float in the range [0, 1]. Possibilities that don't score at least that similar to *word* are ignored.
- The best (no more than) matches among the possibilities are returned in a list, sorted by similarity score, most similar first.
- The idea is to find the longest contiguous matching sub sequence that contains no junk elements these junk items are such as whitespaces or blank lines.
- The context diff format normally has a header for filenames and modification times.
- Unified diffs are a compact way of showing just the lines that have changed plus a few lines of context.

CHAPTER 4

RESULTS AND DISCUSSIONS

This project main motto is that to get the meanings of the entered words that are present in the jsonfile. The attached screenshots represent the different times execution of the program in all possible outcomes of the result.

4.1 SCREENSHOTS



```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\project\appl.py =====
Enter the word rain
Precipitation in the form of liquid water drops with diameters greater than 0.5 millimetres.
To fall from the clouds in drops of water.
Enter the word |
```

Fig. 4.1 Sample output-1

The above diagram describes If a correct word is entered that is present in the json file it will directly display the meaning of the word.

```

Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\project\appl.py =====
Enter the word rainnn
IS rain that letter you are thinking?for yes type 'y' For no type 'n' y
Precipitation in the form of liquid water drops with diameters greater than 0.5 millimetres.
To fall from the clouds in drops of water.
Enter the word

```

Fig. 4.2 Sample output-2

The above diagram describes if a wrong word is entered it checks for the closely related words in the json file and display it. If that is the word print 'y' else 'n'

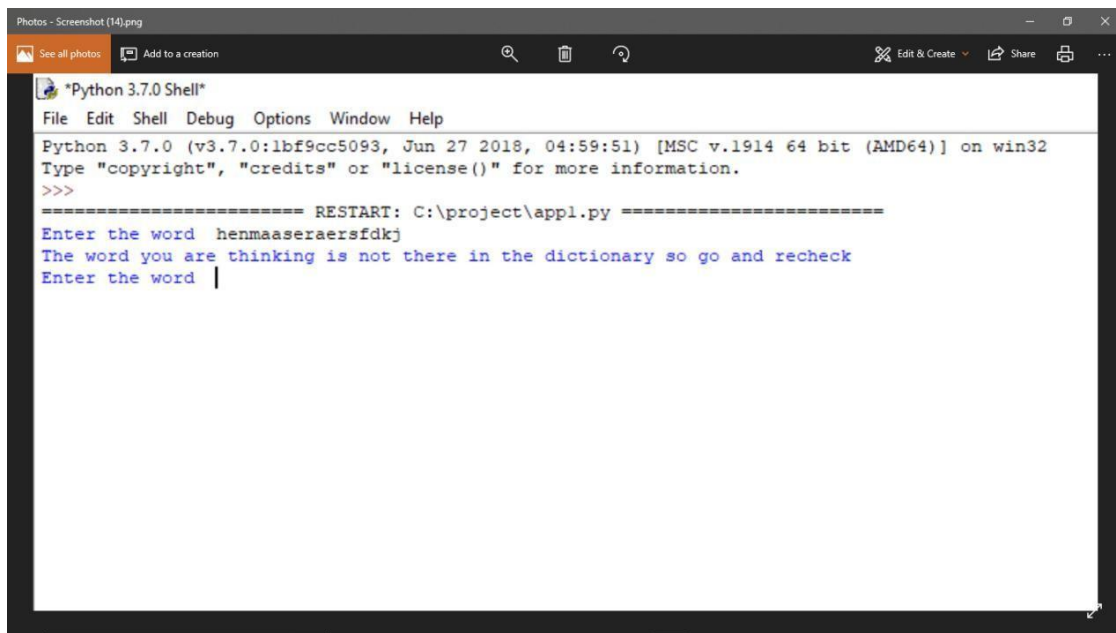
```

Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\project\appl.py =====
Enter the word rainnn
IS rain that letter you are thinking?for yes type 'y' For no type 'n' n
The words related this word are :['train', 'rainy'] For yes type 'y' else type 'n'y
Then now select the index of the word from the list that you are thinking 1
A self-propelled, connected group of vehicles moving on rails.
To point or cause to go (blows, weapons, or objects such as photographic equipment) towards
To develop behaviour by instruction and practice.
A group of animals, vehicles, or people that follow one another in a line.
To do physical exercise to improve one's fitness.
To exercise in order to prepare for an event or competition.
To teach by training.
To undergo training or instruction for a particular role, function, or profession.
To educate for a future role or function.
To act as a trainer or coach (to), as in sports.
Enter the word |

```

Fig. 4.3 Sample output-3

The above diagram describes although it shows the closest matching word. If it is not the word that you are thinking then enter 'n'.If 'n' is entered then it shows the another possible words that are that closely matches to the entered word if the word present in the list then type'y' else'n'.if'y' is typed then enter the index of that word.



```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\project\appl.py =====
Enter the word henmaaseraersfdkj
The word you are thinking is not there in the dictionary so go and recheck
Enter the word |
```

Fig. 4.4 Sample output-4

The above diagram describes if the wrong word is entered that is not at all present in the dictionary then it will display message word not found.

CHAPTER 5

CONCLUSION AND FUTURE WORK

This project is useful for the users who want to know the meaning of the particular word. Even though user entered wrong word it shows the similar words related. It can also be developed by adding the GUI'S and the making use of the all other pre-defined modules in the python like Tkinter, pandas, and can also be developed by using HTML,CSS,JQUERY and JAVASCRIPT and make more useful by running in the HTML page without running in python ide.

REFERENCES

Books

1. Steven Kelly (2010). Domain specific modeling enabling full code generation
2. Peter Fritzson (2015). Principles of object oriented modeling and simulation with Modelica 3.3: A cyber physical approach.
3. Automate the boring stuff with python (2015). A practical programming approach.

Websites

1. https://onlinecourses.nptel.ac.in/noc18_cs21/course
2. <https://automatetheboringstuff.com/>
3. <https://www.udemy.com/>
4. <https://www.javatpoint.com/python-tutorial>
5. <https://stackoverflow.com/questions/8822300/online-python-tutorials>
6. <https://www.w3schools.com/python/>

APPENDIX

B) SOURCECODE:

```
import json
from difflib import get_close_matches
data = json.load(open("data.json"))
def check(w):
    w = w.lower()
    if w in data: # This is executed when the word is entered directly return data[w]
        elif w.title() in data: # if the first letter is capital then this will return the meaning of
            the word
            return data[w.title()]
        elif w.upper() in data: # If the all entered letters are capital then this will return
            meaning of the word
            return data[w.upper()]
        elif len(get_close_matches(w, data.keys())) > 0: # This condition matches the
            closet word to our currently entered wrong word in the dictionary.
                yn = input("IS %s that letter you are thinking? for yes type 'y' For no type 'n'
                " % get_close_matches(w, data.keys())[0])
            if yn == 'y':
                return data[get_close_matches(w, data.keys())[0]]
            elif yn == 'n':
                i = input("The words related this word are :%s For yes type 'y' else type
                'n'" % get_close_matches(w, data.keys())[1:])
            if i == 'y':
                k = int(input("Then now select the index of the word from the list
                that you are thinking "))
                return data[get_close_matches(w, data.keys())[int(k)]]
            elif i == 'n':
                print('The word you are thinking is not there in dictionary')
                k = input(("If you want to check another word then type Y else N"))
                if k == 'y':
```

```

w=input("Enter the word ")

return check(w)
else:
return "please enter a valid Entry of query"
else:
return "Please enter a valid entry of query"
else:
return "The word you are thinking is not there in the dictionary so go and
recheck"

while (True):
w=input('Enter the word ')
output=check(w)
if type(output)==list:
for
item in output:
print(item)
else:
print(output)

```

