

**REPORT**

**On**

**WEATHER APP USING DJANGO**

Submitted In Partial Fulfillment of the Requirements

forthe Degreeof

**DIPLOMA**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted By:**

**BALRAM(200150800007)**

**BATCH: 2020-2023**

**To**

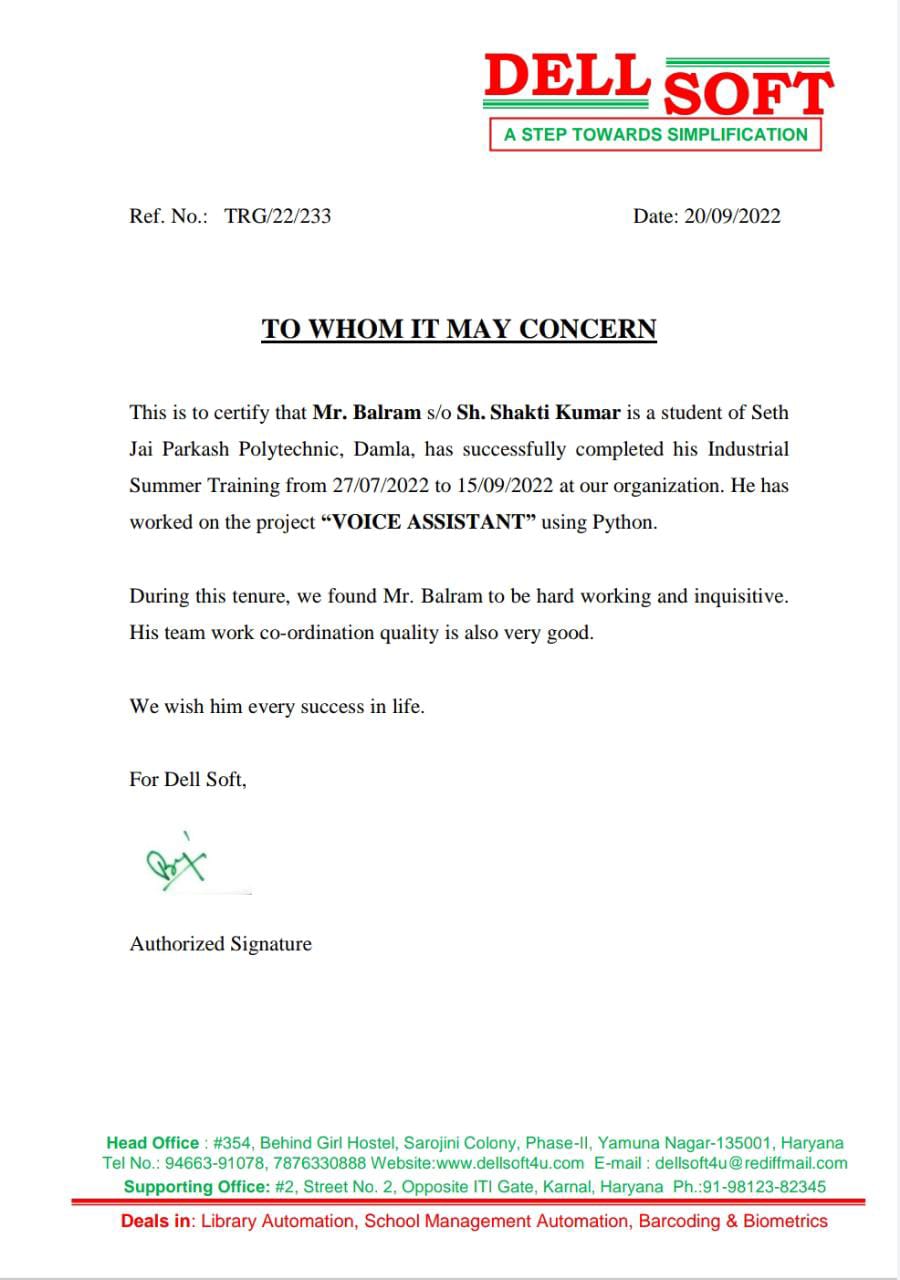
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Seth Jai Parkash Polytechnic,**

**Damla– 135001 (Yamuna Nagar)**

**Affiliated to HSBTE, Panchkula**

**Session:2020-2023**



**Declaration**

I hereby certify that the work which is being presented in the Training Report entitled, **“PYTHON”** byme,**BALRAM(200150800007)**in partialfulfillmentoftherequirementsfortheaward ofdegreeof**DIPLOMA**in **Computer Science &Engineering**

At**SJPP,DAMLA(Affiliated to HSBTE,PUNCHKULA)** is an authentic record of my own work carried out under the supervisionof **Prof./Er. BrijPal Kamboj**

**BALRAM**

**2000150800007**

**Acknowledgement**

The writing of this training report has been assisted by the generous help of many people. I feelthatIwasveryfortunatetoreceiveassistancefromthem.Iwishtoexpressmysincereappreciationtothem.

Firstandforemost,Iamindebtedtomyprincipalsupervisor,**Er./Mr.BrijPal Kamboj**of **DELLSOFT Company** who has been verysupportive atevery stage of my preparations. I wish to express my utmostgratitude tohim/herfor his/her invaluable advice and patience in reading, correcting and commenting on the drafts ofthe report and, more importantly, for his/her generosity which I have received throughout mytrainingprogram.

I would like to acknowledge and extended my heartfelt gratitude to**Sh. BHOOPENDRA SINGH**, Head ofDepartmentwho providedme the golden opportunity to undergo and complete thistrainingprogram.

Finally, I am particularly indebted to my dearest parents/guardians as without their generousassistance andlove;thisdissertationcouldneverhavebeencompleted.

**BALRAM**

**200150800007**

**PREFACE**

This basis for this research originally stemmed from my passion for developing my coding skills.As the world moves further into digital age, generating vast amount of data and born digital content, there will be a greater need to access legacy material created with outdated technology. In truth, I could not have achieved my current level of success without a strong support group. First of all,my parents who supported me with love and understanding and secondly,my committee members, each of whom has provided patient advice and guidance throughout the research process.

Thank you all for your unwavering support.

**CompanyProfile**

DELLSOFT a step towards simplification, Yamuna Nagar is providing career-oriented education to the students so as to equip them with requisite qualification and training.This will enable them either to take up a gainful employment or start their own enterprise andbecome self-sufficient.

DELLSOFT a step towards simplification is totally alive to the needs and aspirations of the studentcommunity and keeps itself abreast of the changes taking place on the industrial front of thecountry and excelling in their field by providing industry-oriented trainings in the area of Object-orientedProgrammingthroughJAVA, PYTHON, WEB DEVELOPMENTand organizations help in improving the quality and gives wider exposure tothe students.

The Company has good market rapport. TheCompany is committed to provide quality vocational and technical education to itsstudents, so as to boost their morale and instill new confidence to prepare them to accept thechallenges of life and put their best foot forward in the march to success for betterment of theirlot.

**CONTENT**

1. INTRODUCTION
2. HISTORY OF PYTHON
3. FEATURES OF PYTHON
4. DATA TYPES
5. OPERATORS
6. FUNCTIOS
7. LOOPS
8. EXCEPTION HANDLING
9. DATABASES
10. STRINGS,MODULES, GUI
11. PROJECT
12. CONCLUSION

**Introduction of Python**

**Python** is a high-level, interpreted scripting language developed in the late 1980s by Guido van Rossum at the National Research Institute for Mathematics and Computer Science in the Netherlands. The initial version was published at the alt. sources [newsgroup](https://en.wikipedia.org/wiki/Usenet) in 1991, and version 1.0 was released in 1994.

Python 2.0 was released in 2000, and the 2.x versions were the prevalent releases until December 2008. At that time, the development team made the decision to release version 3.0, which contained a few relatively small but significant changes that were not backward compatible with the 2.x versions. Python 2 and 3 are very similar, and some features of Python 3 have been backported to Python 2. But in general, they remain not quite compatible.

**History of Python**

## Pythonis a widely used general-purpose, high-level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code. **Let’s dig deeper –** In the late 1980s, history was about to be written. It was that time when working on Python started. Soon after that, Guido Van Rossum began doing its application-based work in December of 1989 by at Centrum Wiskunde & Informatica (CWI) which is situated in Netherland. It was started firstly as a hobby project because he was looking for an interesting project to keep him occupied during Christmas. The programming language which Python is said to have succeeded is ABC Programming Language, which had the interfacing with the Amoeba Operating System and had the feature of exception handling. He had already helped to create ABC earlier in his career and he had seen some issues with ABC but liked most of the features. After that what he did as really very clever. He had taken the syntax of ABC, and some of its good features. It came with a lot of complaints too, so he fixed those issues completely and had created a good scripting language which had removed all the flaws. The inspiration for the name came from BBC’s TV Show – ‘Monty Python’s Flying Circus’, as he was a big fan of the TV show and also, he wanted a short, unique and slightly mysterious name for his invention and hence he named it Python. The language was finally released in 1991. When it was released, it used a lot fewer codes to express the concepts, when we compare it with Java, C++ & C. Its design philosophy was quite good too. Its main objective is to provide code readability and advanced developer productivity. When it was released, it had more than enough capability to provide classes with inheritance, several core data type exception handling and functions.

Feature of python

**EasytoCode**is a very developer-friendly language which means that anyone and everyone can learn to code it in a couple of hours or days. As compared to other object-oriented programming languages like Java, C, C++, and C#, Python is one of the easiest to learn.

**OpenSource**Python is an open-source programming language which means that anyone  
can create and contribute to its development. Python has an online forum  
where thousands of coders gather daily to improve this language further.  
Along with this Pythonis free to download and use in any operating system, be it Windows, Mac or Linux.

**SupportforGUI**GUI or Graphical User Interface is one of the key aspects of any programming language because it has the ability to add flair to code and make the results more visual. Python has support for a wide array of GUIs which can easily be imported to the interpreter, thus making this one of the most favourite languages for developers.

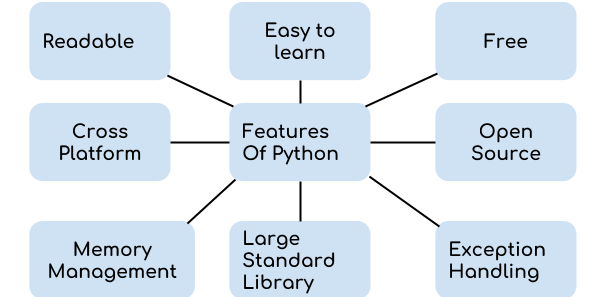
**Object-OrientedApproach**One of the key aspects of Python is its object-oriented approach. This basically means that Python recognizes the concept of class and object encapsulation thus allowing programs to be efficient in the long run.

**High-Level-Language**Python has been designed to be a high-level programming language, which means that when you code in Python you don’t need to be aware of the coding structure, architectureas well as memory management.

**IntegratedbyNature**Python is an integrated language by nature. This means that the python interpreter executes codes one line at a time. Unlike other object-oriented programming languages, we don’t need to compile Python code thus making the debugging process much easier and efficient. Another advantage of this is, that upon execution the Python code is immediately converted into an intermediate form also known as byte-code which makes it easier to execute and also saves

**Portable**Suppose you are running Python on Windows and you need to shift the same to either a Mac or aLinux system, then you can easily achieve the same in Python without having to worry about changing the code. This is not possible in other programming languages, thus making Python one of the most portable language

**Dynamic**As mentioned in an earlier paragraph, Python is one of the most dynamic languages available in the industry today. What this basically means is that the type of a variable is decided at the runtime and not in advance. Due to the presence of this feature, we do not need to specify the type of the variable during coding, thus saving time and increasing efficiency.



**Introduction to Data types**

In computer science and computer programming, a data type or simply type is a classification of data which tells the compiler or interpreter how the programmer intends to use the data. Most programming languages support various types of data, for example: real, integer or Boolean. A data type provides a set of values from which an expression (i.e., variable, function...) may take its values. This data type defines the operations that can be done on the data, the meaning of the data, and the way values of that type can be stored. A type of value from which an expression may take its value

**Variables**

Variables are nothing but reserved memory locations to store values. It means that when you create a variable, you reserve some space in the memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to the variables, you can store integers, decimals or characters in these variables.

**Assigning Values to Variables**

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable.

**Multiple Assignment**

Python allows you to assign a single value to several variables simultaneously.

For example-

a=b=c=1

Here, an integer object is created with the value 1, and all the three variables are assigned to the same memory location. You can also assign multiple objects to multiple variables.

For example-

a, b, c = 1, 2, "john"

Here, two integer objects with values 1 and 2 are assigned to the variables a and b respectively, and one string object with the value "john" is assigned to the variable c.

**Standard Data Types**

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types-

* Numbers
* String
* List
* Tuple
* Dictionary

**Python Numbers**

Number data types store numeric values. Number objects are created when you assign a value to them. For example-

var1 = 1

var2 = 10

You can also delete the reference to a number object by using the del statement. The syntax of the del statement is −

del var1[, var2[, var3[...., varN]]]]

You can delete a single object or multiple objects by using the del statement.

For example-

del var

del vara, varb

Python supports three different numerical types -

* int (signed integers)
* float (floating point real values)
* complex (complex numbers)

All integers in Python 3 are represented as long integers. Hence, there is no separate number type as long.

**Python Strings**

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows either pair of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:] ) with indexes starting at 0 in the beginning of the string and workingtheir way from -1 to the end.

The plus (+) sign is the string concatenation operator and the asterisk (\*) is the repetition operator.

**Data Type Conversion**

Sometimes, you may need to perform conversions between the built-in types. To convert between types, you simply use the type-name as a function.

There are several built-in functions to perform conversion from one data type to another. These functions return a new object representing the converted value.

* **Int(x [,base]):**Converts x to an integer. The base specifies the base if x is a string.
* **float(x):**Converts x to a floating-point number.
* **complex (real [,imag]):** Creates a complex number.
* **str(x):** Converts object x to a string representation.
* **repr(x):** Converts object x to an expression string.
* **eval(str)** Evaluates a string and returns an object.
* **tuple(s):** Converts s to a tuple.
* **list(s):** Converts s to a list.
* **set(s):** Converts s to a set.
* **dict(d):** Creates a dictionary. d must be a sequence of (key,value) tuples.
* **frozenset(s):** Converts s to a frozen set.
* **chr(x):** Converts an integer to a character.
* **unichr(x):** Converts an integer to a Unicode character.
* **ord(x):** Converts a single character to its integer value.
* **hex(x):** Converts an integer to a hexadecimal string.
* **oct(x):** Converts an integer to an octal string.

**Operators**

Operators are the constructs, which can manipulate the value of operands. Consider the expression 4 + 5 = 9. Here, 4 and 5 are called operands and + is called the operator.

**Types of Operators**

Python language supports the following types of operators-

* Arithmetic Operators
* Comparison (Relational) Operators
* Assignment Operators
* Logical Operators
* Bitwise Operators
* Membership Operators
* Identity Operators

## Python Comparison Operators

These operators compare the values on either side of them and decide the relation among them. They are also called Relational operators.

Assume variable a holds the value 10 and variable b holds the value 20, then-

* **==**

If the values of two operands are equal, then the condition becomes true.

* **!=**

If values of two operands are not equal, then condition becomes true.

* **>**

If the value of left operand is greater than the value of right operand, then condition becomes true.

**. <** If the value of left operand is less than the value of right operand, then condition becomes true.

* **>=**

If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.

* **<=**

If the value of left operand is less than or equal to the value of right operand, then condition becomes true.

**Python Logical Operators**

The following logical operators are supported by Python language. Assume variable a holds True and variable b holds False then-

* **and Logical AND**

If both the operands are true then condition becomes true.

example:-

(a and b) is False.

* **or Logical OR**

If any of the two operands are non-zero then condition becomes true.

example:-

(a or b) is True.

* **not Logical NOT**

## **Python Bitwise Operators**

Bitwise operator works on bits and performs bit-by-bit operation. Assume if a = 60; and b = 13; Now in binary format they will be as follows-

a = 0011 1100

b = 0000 1101

-----------------

a&b = 0000 1100

a|b = 0011 1101

a^b = 0011 0001

~a = 1100 0011

Python's built-in function bin() can be used to obtain binary representation of an integer number.

The following Bitwise operators are supported by Python language-

* **& Binary AND**

Operator copies a bit to the result, if it exists in both operands

example:-

(a & b) (means 0000 1100)

* **| Binary OR**

It copies a bit, if it exists in either operand.

example:-

* **^ Binary XOR**

It copies the bit, if it is set in one operand but not both.

example:-

(a ^ b) = 49 (means 0011 0001)

* **~ Binary Ones Complement**

It is unary and has the effect of 'flipping' bits.

example:-

(~a ) = -61 (means 1100 0011 in 2's complement form due to a signed binary number.

* **<< Binary Left Shift**

The left operand’s value is moved left by the number of bits specified by the right operand.

example:-

a << = 240 (means 1111 0000)

* **>> Binary Right Shift**

The left operand’s value is moved right by the number of bits specified by the right operand.

example:-

**Python Membership Operators**

Python’s membership operators test for membership in a sequence, such as strings, lists, or tuples. There are two membership operators as explained below-

* **in**

Evaluates to true, if it finds a variable in the specified sequence and false otherwise.

example:-

x in y, here in results in a 1 if x is a member of sequence y.

* **not in**

Evaluates to true, if it does not find a variable in the specified sequence and false otherwise.

example:-

x not in y, here not in results in a 1 if x is not a member of sequence Y

**Python Identity Operators**

There aretwo types of identity operators

.is

.is not

**Functions**

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

As you already know, Python gives you many built-in functions like print(), etc. but you can also create your own functions. These functions are called user-defined functions.

**Defining a Function**

You can define functions to provide the required functionality. Here are simple rules to define a function in Python.

* Function blocks begin with the keyword def followed by the function name and parentheses ( ( ) ).
* Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
* The first statement of a function can be an optional statement - the documentation string of the function or docstring.
* The code block within every function starts with a colon (:) and is indented.
* The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

**Calling a Function**

Defining a function gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.

Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt. Following is an example to call the printme() function-

**Pass by Reference vs Value**

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function.

**The Anonymous Function**

These functions are called anonymous because they are not declared in the standard manner by using thedefkeyword. You can use thelambdakeyword to create small anonymous functions.

* .Lambda forms can take any number of arguments but return just one value in the form of an expression. They cannot contain commands or multiple expressions.
* An anonymous function cannot be a direct call to print because lambda requires an expression.

# **Python - Error Types**

The most common reason of an error in a Python program is when a certain statement is not in accordance with the prescribed usage. Such an error is called a syntax error. The Python interpreter immediately reports it, usually along with the reason.

Many times, though, a program results in an error after it is run even if it doesn't have any syntax error. Such an error is a runtime error, called an exception. A number of built-in exceptions are defined in the Python library. Let's see some common error types.

**IndexError** is thrown when trying to access an item at an invalid index.

**ModuleNotFoundError** is thrown when a module could not be found.

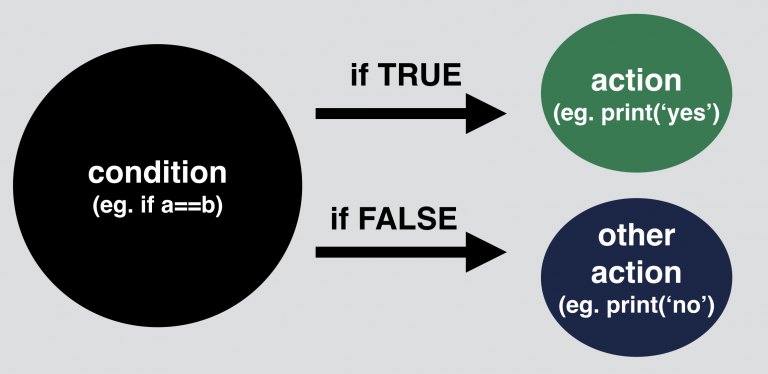
**ImportError** is thrown when a specified function cannot be found.

**Stop Iteration** is thrown when the next () function goes beyond the iterator items.

**TypeError** is thrown when an operation or function is applied to an object of an inappropriate type.

**NameError** is thrown when an object could not be found.**If-Else**

We use if statements in our everyday life all the time - even if our everyday life is not written in Python. If the light is green then I'll cross the road; otherwise I'll wait. If the sun is up then I'll get out of bed; otherwise I'll go back to sleep. Okay, maybe it's not this direct, but when we take actions based on conditions, our brain does what a computer would do: evaluate the conditions and act upon the results. Well, a computer script doesn't have a subconscious mind, so for practicing data science we have to understand how an if statement works and how we can apply it in Python!



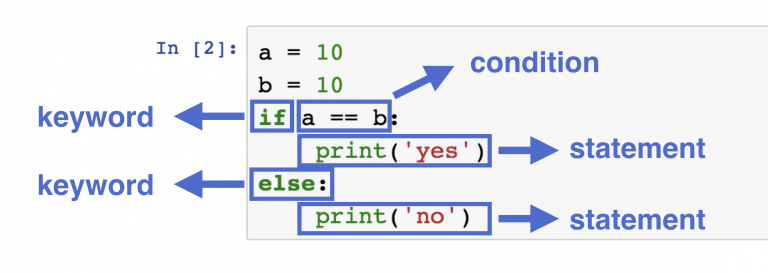
Let's say we have two values: a = 10 and b = 20. We compare these two values:

a == b.  
This comparison has either a True or a False output.

**Python if statement syntax**

Let's take a look at the syntax

The basics are simple:



a =10

b =11

c =10

if a==b:

print(“first condition is true”)

elseif a==c:

print(“second condition is true”)

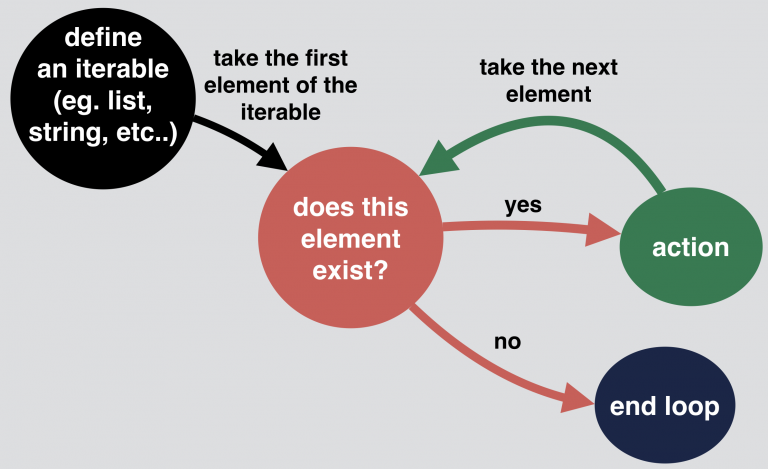
else:

print(“nothing is true”)

second condition is true

# **Loops**

You go through your shopping list until you've collected every item from it. The dealer gives a card for each player until everyone has five.  
The athlete does push-ups until reaching one-hundred... Loops everywhere! As for loops in Python: they are perfect for processing repetitive programming tasks.  
  
In Python, and all other modern programming languages, iteration statements (also called loops) allow a set of instructions to be repeatedly executed until a certain condition is reached. This condition may be predetermined (as in the for loop) or open-ended (as in the while and do-while loops). In another way we can say, A loop can be used to tell a program to execute statements repeatedly. Or we can say that a loop repeatedly executes the same set of instructions until a termination condition is met.



**For Loop**

Python implements an iterator-based 'for loop'. It is a type of 'for loop' that iterates over a list of items through an explicit or implicit iterator.

The loop is introduced by the keyword 'for' which is followed by a random variable name

which will contain the values supplied by the object.

This is the syntax of Python's 'for loop':

for variable in list:  
statements  
else:  
statements

**The While Loop**

A Python 'while loop' repeatedly carries out a target statement while the condition is true. The loop iterates as long as the defined condition is true. When it ceases to be true and becomes false, control passes to the first line after the loop.

The ‘While’loop ha the following syntax:

while condition  
statement  
statement

Here is a simple 'while loop':

counter = 0  
while (counter < 10):  
print('The count is:' , counter)  
counter = counter + 1  
print('Done!')

**The Python Standard Library**

While [The Python Language Reference](https://docs.python.org/3/reference/index.html#reference-index) describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions.

Python’s standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

The Python installers for the Windows platform usually include the entire standard library and often also include many additional components. For Unix-like operating systems Python is normally provided as a collection of packages, so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components.

In addition to the standard library, there is a growing collection of several thousand components (from individual programs and modules to packages and entire application development frameworks), available from the [Python Package Index](https://pypi.org/).

# **Python Exception**

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 provides a way to handle the exception so that the code can be executed without any interruption. If we do not handle the exception, the interpreter doesn't execute all the code that exists after the exception.

**Common Exceptions**

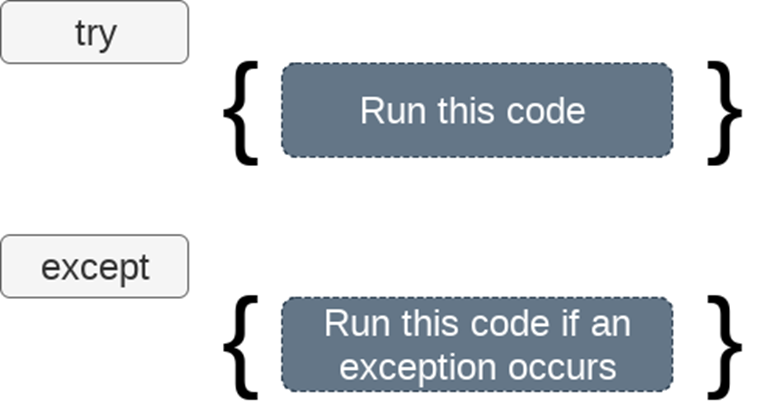
Python provides the number of built-in exceptions, but here we are describing the common standard exceptions. A list of common exceptions that can be thrown from a standard Python program is given below.

1. **ZeroDivisionError:** Occurs when a number is divided by zero.
2. **NameError:** It occurs when a name is not found. It may be local or global.
3. **IndentationError:** If incorrect indentation is given.
4. **IOError:** It occurs when Input Output operation fails.
5. **EOFError:** It occurs when the end of the file is reached, and yet operations are being performed.

**Exception handling in python**

The try-expect statement

If the Python program contains suspicious code that may throw the exception, we must place that code in the **try** block. The **try** block must be followed with the **except** statement, which contains a block of code that will be executed if there is some exception in the try block.



**Databases**

A database is an abstraction over an [operating system](https://www.fullstackpython.com/operating-systems.html)'s file system that makes it easier for developers to build applications that create, read, update and delete persistent data.

**Why are databases necessary?**

At a high-level web application store data and present it to users in a useful way. For example, Google stores data about roads and provides directions to get from one location to another by driving through the [Maps](https://www.google.com/maps/) application. Driving directions are possible because the data is stored in a structured format.

Databases make structured storage reliable and fast. They also give you a mental framework for how the data should be saved and retrieved instead of having to figure out what to do with the data every time you build a new application.

**Most common databases for Python web apps**

[PostgreSQL](http://www.postgresql.org/) and [MySQL](http://www.mysql.com/) are two of the most common open-source databases for storing Python web applications' data.

[SQLite](http://www.sqlite.org/) is a database that is stored in a single file on disk. SQLite is built into Python but is only built for access by a single connection at a time. Therefore, is highly recommended to not [run a production web application with SQLite](https://docs.djangoproject.com/en/dev/ref/databases/#database-is-locked-errors).

**PostgreSQL database**

PostgreSQL is the recommended relational database for working with Python web applications. PostgreSQL's feature set, active development and stability contribute to its usage as the backend for millions of applications live on the Web today.

Learn more about using PostgreSQL with Python on the [PostgreSQLpage](https://www.fullstackpython.com/postgresql.html).

**MySQL database**

MySQL is another viable open-source database implementation for Python applications. MySQL has a slightly easier initial learning curve than PostgreSQL but is not as feature rich.

Find out about Python applications with a MySQL backed on the dedicated [MySQL page](https://www.fullstackpython.com/mysql.html).

**STRINGS**

Strings are amongst the most popular types in Python. We can create them simply by enclosing characters in quotes. Python treats single quotes the same as double quotes. Creating strings is as simple as assigning a value to a variable. For example −

var1 ='Hello World!'

var2 ="Python Programming"

**Accessing Values in Strings**

Python does not support a character type; these are treated as strings of length one, thus also considered a substring.

To access substrings, use the square brackets for slicing along with the index or indices to obtain your substring. For example −

#!/usr/bin/python

var1 ='Hello World!'

var2 ="Python Programming"

print"var1[0]: ", var1[0]

print"var2[1:5]: ", var2[1:5]

When the above code is executed, it produces the following result −

var1[0]: H

var2[1:5]: ytho

**MODULES**

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.

Example

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here's an example of a simple module, support.py

defprint\_func( par):

print"Hello : ", par

return

**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[... moduleN]

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module. For example, to import the module support.py, you need to put the following command at the top of the script

A module is loaded only once regardless the time it is imported This prevents the module execution from happening over and over

**GRAPHICAL USER INTERFACE**

Python provides various options for developing graphical user interfaces (GUIs). Most important are listed below.

* **Tkinter** − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.
* **wxPython** − This is an open-source Python interface for wxWindows [http://wxpython.org](http://wxpython.org/).
* **JPython** − JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine [http://www.jython.org](http://www.jython.org/).

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

Tkinter Programming

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

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

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

**PROJECT OF PYTHON**

**VOICE ASSISTANT :**

It is a software that carries out everyday tasks via voice command. The query for the assistant can be manipulated as per the user’s need. This is commonly used in voice assistants like Alexa, Siri, etc. Python provides an API called SpeechRecognition to allow us to convert audio into text for further processing

**Modules needed**:

* Pyttsx3:- This module is used for the conversion of text to speech in a program it works offline.
* Wikipedia:- As we all know Wikipedia is a great source of knowledge just like geeksforgeeks we have used the Wikipedia module to get information from Wikipedia or to perform a Wikipedia search
* Speech Recognition:- Since we’re building an Application of voice assistant, one of the most important things in this is that your assistant recognizes your voice (means what you want to say/ ask).
* Pyjokes:-Pyjokes is used for the collection of Python Jokes over the Internet
* BeautifulSoup: Beautiful Soup is a library that makes it easy to scrape information from web pages
* Requests: Requests is used for making GET and POST requests.
* Playsound: The playsound module contains only one thing - the function (also named) playsound.It requires one argument - the path to the file with the sound you’d like to play

**SOURCE CODE**

import datetime

import time

importwebbrowser

importos

import pyttsx3

importrequests , json

importspeech\_recognitionassr

importwikipedia

importpyjokes

import requests

from bs4 importBeautifulSoup

fromplaysoundimportplaysound

engine = pyttsx3.init()

voices = engine.getProperty('voices')

engine.setProperty('voice', voices[1].id)

def speak(text):

    engine.say(text)

    engine.runAndWait()

defwishMe():

    playsound('abc.mp3')

    hour = datetime.datetime.now().hour

    if0<= hour <12:

        speak("Hello,Good Morning Sir ")

        print("Hello,Good Morning Sir ")

    elif12<= hour <18:

        speak("Hello,Good Afternoon Sir ")

        print("Hello,Good Afternoon Sir ")

    else:

        speak("Hello,Good Evening Sir ")

        print("Hello,Good Evening Sir ")

deftakeCommand():

    r = sr.Recognizer()

    withsr.Microphone() as source:

        print("Listening...")

        r.adjust\_for\_ambient\_noise(source)

        audio = r.listen(source)

        try:

            state = r.recognize\_google(audio)

            print("User said:{}\n".format(state))

        except :

            speak("Sorry, please say that again")

            return"None"

        return state

print("Loading your AI personal assistant friday")

speak("Loading your AI personal assistant friday")

wishMe()

if \_\_name\_\_ == '\_\_main\_\_':

    whileTrue:

        speak("Tell me how can I help you now?")

        statement = takeCommand().lower()

        if statement == 0:

            continue

        if"good bye"in statement or"ok bye"in statement or"stop"in statement:

            speak('your personal assistant friday is shutting down,Good bye')

            print('Your personal assistant friday is shutting down,Good bye')

            break

        elif'how are you'in statement:

            speak("I am fine, Thank you")

            speak("How are you, Sir")

        elif'fine'in statement or"good"in statement:

            speak("It's good to know that your fine")

            time.sleep(5)

        elif"what's your name"in statement or"What is your name"in statement:

            speak("My friends call me")

            speak('friday')

            print("My friends call me FRIDAY")

            time.sleep(5)

        elif'is love'in statement:

            speak("It is 7th sense that destroy all other senses")

            time.sleep(5)

        elif"who are you"in statement:

            speak("I am your virtual assistant created by Balram")

            time.sleep(5)

        elif'joke'in statement:

            joke=pyjokes.get\_joke()

            speak('joke')

            print('joke')

            time.sleep(5)

        elif'wikipedia'in statement:

            speak('Searching Wikipedia...')

            statement = statement.replace("wikipedia", "")

            results = wikipedia.summary(statement, sentences=3)

            speak("According to Wikipedia")

            print(results)

            speak(results)

            time.sleep(5)

        elif'open youtube'in statement or'youtube'instatement :

            webbrowser.open\_new\_tab("https://www.youtube.com")

            speak("Opening YOUTUBE ")

            time.sleep(5)

        elif'open google'in statement or'google chrome'in statement or'open browser'in statement:

            webbrowser.open\_new\_tab("https://www.google.com")

            speak("Default Brouser is opening now")

            time.sleep(5)

        elif'open gmail'in statement:

            webbrowser.open\_new\_tab("https://mail.google.com/mail/u/0/")

            speak("Opening Mail Services ")

            time.sleep(5)

        elif'time'in statement:

            strTime = datetime.datetime.now().strftime("%H:%M:%S")

            speak(f"The Time is {strTime}")

            time.sleep(6)

        elif"don't listen"in statement or"pause"in statement:

            speak("friday is sleeping now....... See You SOON sir")

            print("friday is sleeping now.......")

            a=input("Enter Any KEY TO Awake")

            continue

        elif "where is" in statement or "location" in statement or "search location" in statement :

            if "where is" in statement:

                 statement = statement.replace("where is", "")

            elif "location" in statement:

                statement = statement.replace("location", "")

            elif "search location" in statement:

                statement = statement.replace("search location", "")

            location = str(statement)

            speak("User asked to Locate")

            speak(location)

            webbrowser.open("https://www.google.co.in/maps/search/"+location)

            time.sleep(5)

        elif "will you be my gf" in statement or "will you be my bf"in statement:

            speak("I'm not sure about, may be you should give me some time")

            print("I'm not sure about, may be you should give me some time")

            time.sleep(3)

        elif "how are you" in statement:

            speak("I'm fine, glad you me that")

            print("I'm fine, glad you me that")

            time.sleep(3)

        elif "i love you" in statement:

            speak("It's hard to understand")

            print("It's hard to understand")

            time.sleep(5)

        elif 'search' in statement:

            statement = statement.replace("search", "")

            assert isinstance(statement)

            assert isinstance(statement)

            webbrowser.open\_new\_tab(statement)

            time.sleep(5)

        elif 'weather' in statement:

            city = "yamunanagar,haryana"

            url = "https://www.google.com/search?q="+"weather"+city

            html = requests.get(url).content

            soup = BeautifulSoup(html, 'html.parser')

            temp = soup.find('div', attrs={'class': 'BNeawe iBp4i AP7Wnd'}).text

            str = soup.find('div', attrs={'class': 'BNeawe tAd8D AP7Wnd'}).text

            data = str.split('\n')

            time = data[0]

            sky = data[1]

            listdiv = soup.findAll('div', attrs={'class': 'BNeawe s3v9rd AP7Wnd'})

            strd = listdiv[5].text

            pos = strd.find('Wind')

            other\_data = strd[pos:]

            print("Temperature is", temp)

            speak("Temperature is"+temp)

            print("Time: ", time)

            speak("Time: "+time)

            print("Sky Description: ", sky)

            speak("Sky Description: "+sky)

            print(other\_data)

            speak(""+other\_data)

        elif'spotify'in statement or'play music'in statement or'open spotify'in statement:

            speak("opening spotify please login first to listen music")

            webbrowser.open\_new\_tab("https://open.spotify.com/playlist/37i9dQZF1DXcBWIGoYBM5M")

            time.sleep(5)

# **Output:-**

