TRAINING – REPORT - FILE

On

YOUTUBE VIDEO DOWNLOADER APPLICATION

Submitted to MAHARAJA RANJIT SINGH PUNJAB TECHNICAL UNIVERSITY in partial fulfillment of the requirement for the award of the degree of

B.TECH

in

COMPUTER SCIENCE & ENGINEERING

Submitted By

MAYANK KUMAR

(170280312)

AKSHAT SRIVASTAVA.

(99170280151)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

BABA HIRA SINGH BHATTAL INSTITUTE OF ENGINEERING & TECHNOLOGY, LEHRAGAGA, SANGRUR-148031

[AUG-DEC(2020)]

ACKNOWLEDGEMENT

It is great pleasure to present this report on the project named “YOUTUBE VIDEO DOWNLOADER APPLICATION” undertaken by me as part of my MCA curriculum.
I am thankful to MRSPTU and COLLEGE for offering me such a wonderful challenging opportunity and I express my deepest thanks to all coordinators, of COLLEGE for providing all the possible help and assistance and their constant encouragement.

It is a pleasure that we find ourselves penning down these lines to express our sincere thanks to the people who helped us along the way in completing our project. We find inadequate words to express our sincere gratitude towards them.

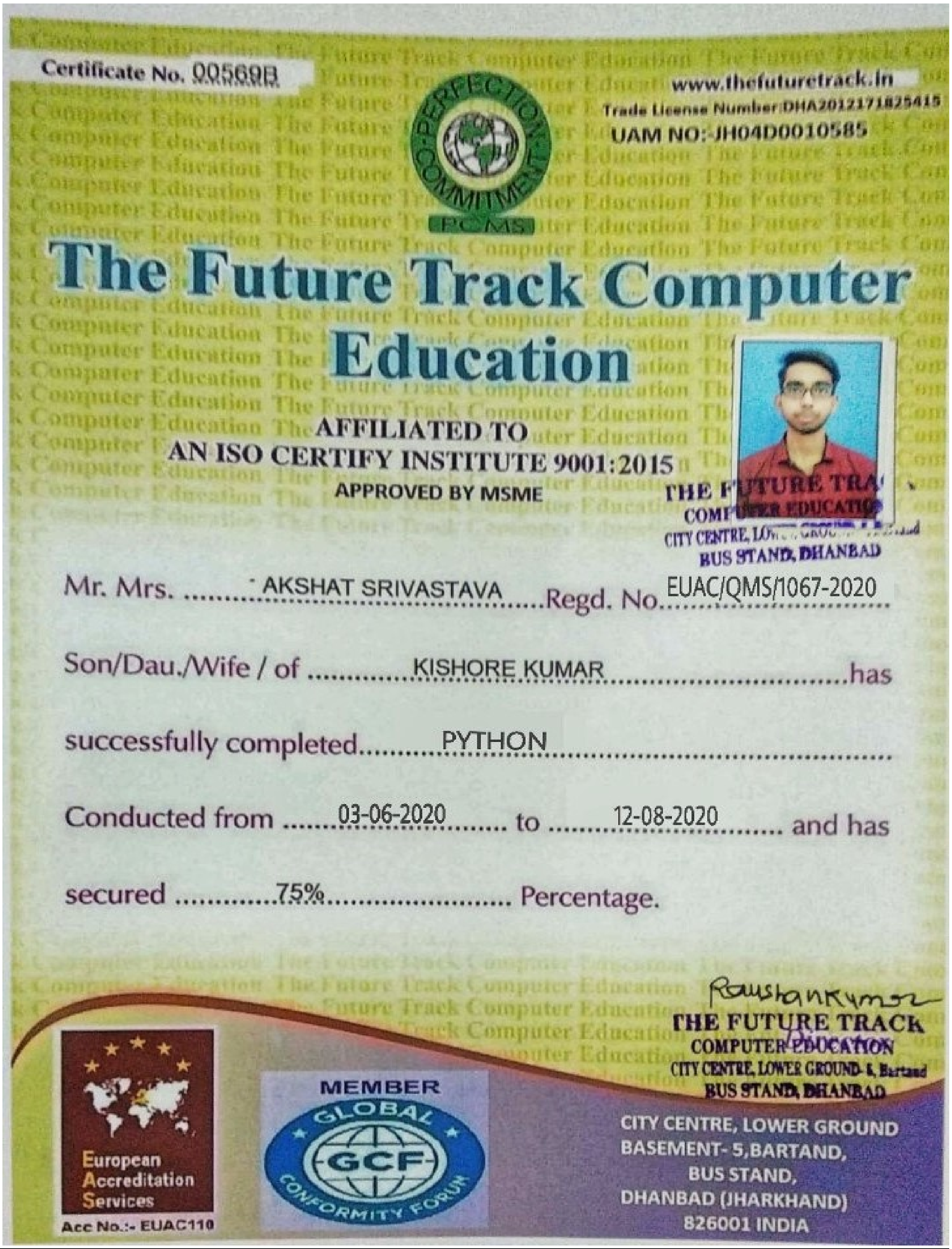
First and foremost we would like to express our gratitude towards our training guide **Mr. Roshan Kumar** for placing complete faith and confidence in our ability to carry out this project and for providing us her time, inspiration, encouragement, help, valuable guidance, constructive criticism and constant interest. She took personal interest in spite of numerous commitments and busy schedule to help us complete this project. Without the sincere and honest guidance of our respected project guide we would have not been to reach the present stage.

Name of Student

Mayank kumar

Akshat srivastava

CERTIFICATE OF TRAINING



Certificate No. 00569B

www.thefuturetrack.in

Trade License Number:DHA2012171825415

UAM NO:-JH04D0010585

PERFECTION COMMITMENT PCMS

The Future Track Computer Education

AFFILIATED TO
AN ISO CERTIFY INSTITUTE 9001:2015
APPROVED BY MSME

THE FUTURE TRACK COMPUTER EDUCATION CITY CENTRE, LOWER GROUND-5, Bartand BUS STAND, DHANBAD

Mr. Mrs. ......AKSHAT SRIVASTAVA......Regd. No. EUAC/QMS/1067-2020

Son/Dau./Wife / of ......KISHORE KUMAR......has

successfully completed......PYTHON......

Conducted from ......03-06-2020...... to ......12-08-2020...... and has

secured ......75%...... Percentage.

Raushankumar

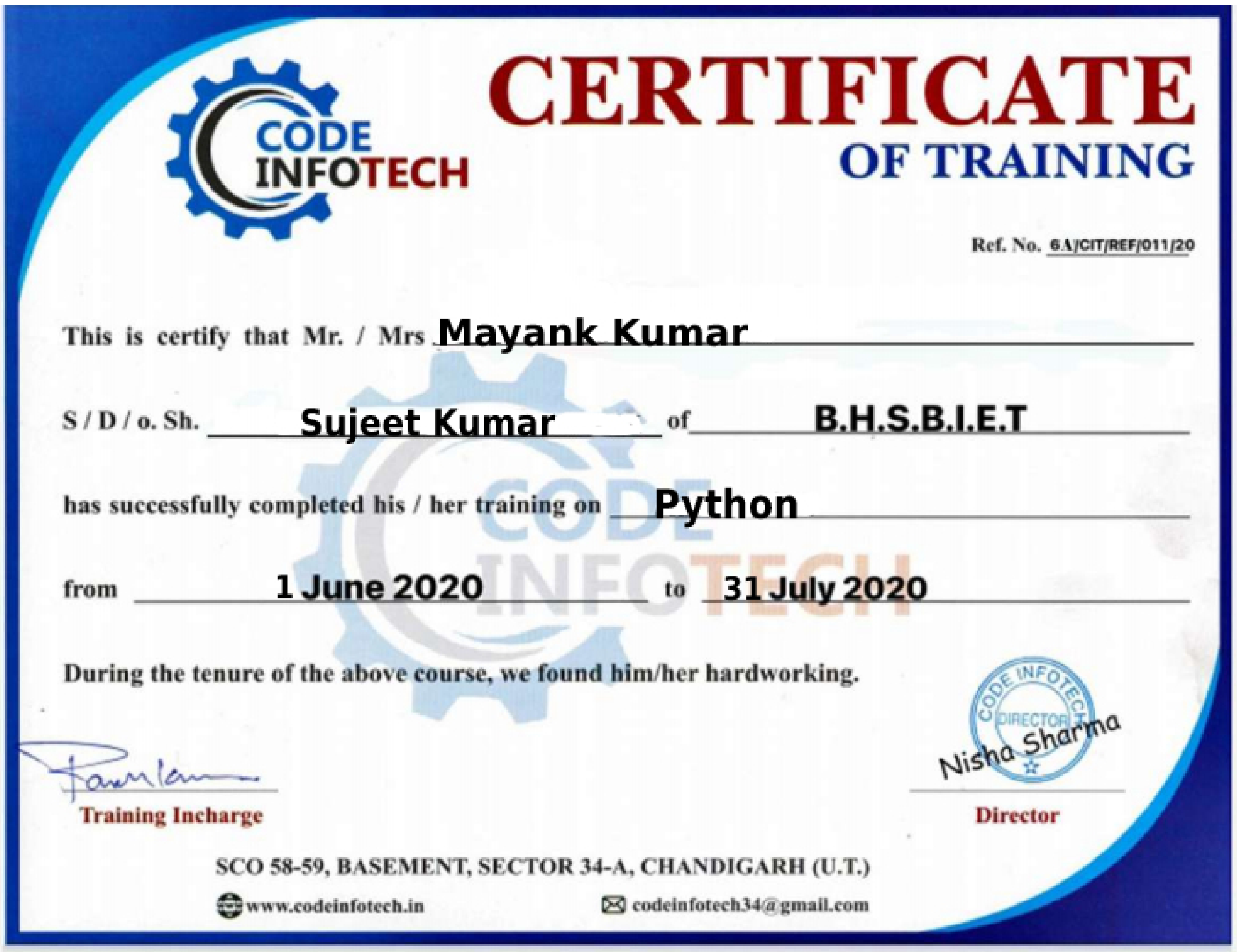
THE FUTURE TRACK COMPUTER EDUCATION CITY CENTRE, LOWER GROUND-5, Bartand BUS STAND, DHANBAD

Director

European Accreditation Services
Acc No.:- EUAC110

MEMBER GLOBAL GCF CONFORMITY FORUM

CITY CENTRE, LOWER GROUND
BASEMENT- 5,BARTAND,
BUS STAND,
DHANBAD (JHARKHAND)
826001 INDIA



CODE INFOTECH

CERTIFICATE OF TRAINING

Ref. No. 6A/CIT/REF/011/20

This is certify that Mr. / Mrs **Mayank Kumar**

S / D / o. Sh. **Sujeet Kumar** of **B.H.S.B.I.E.T**

has successfully completed his / her training on **Python**

from **1 June 2020** to **31 July 2020**

During the tenure of the above course, we found him/her hardworking.

[illegible]

Training Incharge

CODE INFOTECH DIRECTOR

Nisha Sharma

Director

SCO 58-59, BASEMENT, SECTOR 34-A, CHANDIGARH (U.T.)

www.codeinfotech.in codeinfotech34@gmail.com

CANDIDATE'S DECLARATION

I,am Mayank kymar, Roll No. 170280312 Akshat Srivastava, Roll No. 99170280151, B.Tech (Semester-VII) of the **Baba Hira Singh Bhattal Institute of Engineering & Technology, Lehragaga** hereby declare that the Training Report entitled **"YOUTUBE VIDEO DOWNLOADER APPLICATION"** is an original work and data provided in the study is authentic to the best of my knowledge. This report has not been submitted to any other Institute for the award of any other degree.

Name of Student

Mayank kumar

(170280312)

Akshat srivastava

(99170280151)

TABLE OF CONTENTS

CHAPTER 10 : Project (YOUTUBE VIDEO DWONLODER APPLICATION)

Chapter 1
Introduction

What is Python?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

History of Python

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! He was the "Benevolent dictator for life" (BDFL) until he stepped down from the position as the leader on 12th July 2018. For quite some time he used to work for Google, but currently, he is working at Dropbox.

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 types exception handling and functions.

Python 3.9.0 is the latest version.

Python 3.9. 0 is the newest major **release** of the **Python** programming language, and it contains many new features and optimizations.

The very Basics of Python

There are a few features of python which are different than other programming languages, and which should be mentioned early on so that subsequent examples don't seem confusing. Further information on all of these features will be provided later, when the topics are covered in depth. Python statements do not need to end with a special character – the python interpreter knows that you are done with an individual statement by the presence of a newline, which will be generated when you press the "Return" key of your keyboard. If a statement spans more than one line, the safest course of action is to use a backslash (\) at the end of the line to let python know that you are going to continue the statement on the next line; you can continue using backslashes on additional continuation lines. (Thereare situations where the backslashes are not needed which will be discussed later.)

Basic Principles of Python

Python has many features that usually are found only in languages which are much more complex to learn and use. These features were designed into python from its very first beginnings, rather than being accumulated into an end result, as is the case with many other scripting languages. If you're new to programming, even the basic descriptions which follow may seem intimidating. But don't worry – all of these ideas will be made clearer in the chapters which follow. The idea of presenting these concepts now is to make you aware of how python works, and the general philosophy behind python programming. If some of the concepts that are introduced here seem abstract or overly complex, just try to get a general feel for the idea, and the details will be fleshed out later

Basic Core Language

Python is designed so that there really isn't that much to learn in the basic language. For example, there is only one basic structure for conditional programming (if/else/elif), two looping commands (while and for), and a consistent method of handling errors (try/except) which apply to all python programs. This doesn't mean that the language is not flexible and powerful, however. It simply means that you're not confronted with an overwhelming choice of options at every turn, which can make programming a much simpler task.

Modules

Python relies on modules, that is, self-contained programs which define a variety of functions and data types, that you can call in order to do tasks beyond the scope of the basic core language by using the import command. For example, the core distribution of python contains modules for processing files, accessing your computer's operating system and the internet, writing CGI scripts (which handle communicating with pages displayed in web browsers), string handling and many other tasks. Optional modules, available on the Python web site (http://www.python.org), can be used to create graphical user interfaces, communicate with data bases, process image files, and so on. This structure makes it easy to get started with python, learning specific skills only as you need them, as well as making python run more efficiently by not always including every capability in every program.

Object Oriented Programming

Python is a true object-oriented language. The term "object oriented" has become quite a popular buzzword; such high profile languages as C++ and Java are both object oriented by design. Many other languages add some object-oriented capabilities, but were not designed to be object oriented from the ground up as python was. Why is this feature important? Object oriented program allows you to focus on the data you're interested in, whether it's employee information, the results of a scientific experiment or survey, setlists for your favorite band, the contents of your CD collection, information entered by an internet user into a search form or shopping cart, and to develop methods to deal efficiently with your data. A basic concept of object oriented programming is encapsulation, the ability to define an object that contains your data and all the information a program needs to operate on that data. In this way, when you call a function (known as a method in object-oriented lingo), you don't need to specify a lot of details about your data, because your data object "knows" all about itself. In addition, objects can inherit from other objects, so if you or someone else has designed an object that's very close to one you're interested in, you only have to construct those methods which differ from the existing object, allowing you to save a lot of work. Another nice feature of object oriented programs is operator overloading. What this means is that the same operator can have different meanings when used with different types of data. For example, in python, when you're dealing with numbers, the plus sign (+) has its usual obvious meaning of addition. But when you're dealing with strings, the plus sign means to join the two strings together. In addition to being able to use overloading for built-in types (like numbers and strings), python also allows you to define what operators mean for the data types you create yourself. Perhaps the nicest feature of object-oriented programming in python is that you can use as much or as little of it as you want. Until you get comfortable with the ideas behind object-oriented programming, you can write more traditional programs in python without any problems.

Namespaces and Variable Scoping

When you type the name of a variable inside a script or interactive python session, python needs to figure out exactly what variable you're using. To prevent variables you create from overwriting or interfering with variables in python itself or in the modules you use, python uses the concept of multiple namespaces. Basically, this means that the same variable name can be used in different parts of a program without fear of destroying the value of a variable you're not concerned with.
To keep its bookkeeping in order, python enforces what is known as the LGB rule. First, the local namespace is searched, then the global namespace, then the namespace of python built-in functions and variables. A local namespace is automatically created whenever you write a function, or a module containing any of functions, class definitions, or methods. The global namespace consists primarily of the variables you create as part of the "toplevel" program, like a script or an interactive session. Finally, the built-in namespace consists of the objects which are part of python's core.

Exception Handling

Regardless how carefully you write your programs, when you start using them in a variety of situations, errors are bound to occur. Python provides a consistent method of handling errors, a topic often refered to as exception handling. When you're performing an operation that might result in an error, you can surround it with a try loop, and provide an except clause to tell python what to do when a particular error arises. While this is a fairly advanced concept, usually found in more complex languages, you can start using it in even your earliest python programs. As a simple example, consider dividing two numbers. If the divisor is zero, most programs (python included) will stop running, leaving the user back at a system shell prompt, or with nothing at all. Here's a little python program that

Chapter 2
Python Variable

Python Variable

Variable is a name that is used to refer to memory location. Python variable is also known as an identifier and used to hold value. In Python, we don't need to specify the type of variable because Python is a infer language and smart enough to get variable type. Variable names can be a group of both the letters and digits, but they have to begin with a letter or an underscore. It is recommended to use lowercase letters for the variable name. Rahul and rahul both are two different variables.

Identifier Naming

Variables are the example of identifiers. An Identifier is used to identify the literals used in the program. The rules to name an identifier are given below.
The first character of the variable must be an alphabet or underscore (_).
All the characters except the first character may be an alphabet of lower-case(a-z), upper-case (A-Z), underscore, or digit (0-9).
Identifier name must not contain any white-space, or special character (!, @, #, %, ^, &, *).
Identifier name must not be similar to any keyword defined in the language.
Identifier names are case sensitive; for example, my name, and MyName is not the same.
Examples of valid identifiers: a123, _n, n_9, etc.
Examples of invalid identifiers: 1a, n%4, n 9, etc.

Declaring Variable and Assigning Values

Python does not bind us to declare a variable before using it in the application. It allows us to create a variable at the required time.
We don't need to declare explicitly variable in Python. When we assign any value to the variable, that variable is declared automatically.
The equal (=) operator is used to assign value to a variable.

Object References

It is necessary to understand how the Python interpreter works when we declare a variable. The process of treating variables is somewhat different from many other programming languages.
Python is the highly object-oriented programming language; that's why every data item belongs to a specific type of class.

Object Identity

In Python, every created object identifies uniquely in Python. Python provides the guaranteed that no two objects will have the same identifier. The built-in **id()** function, is used to identify the object identifier.

Python Variable Types

There are two types of variables in Python - Local variable and Global variable. Let's understand the following variables.

Local Variable

Local variables are the variables that declared inside the function and have scope within the function. Let's understand the following example.

Example -

```
# Declaring a function
def add():
    # Defining local variables. They has scope only within a function
    a = 20
    b = 30
    c = a + b
    print("The sum is:", c)

# Calling a function
add()
```

Output:

```
The sum is: 50
```

Global Variables

Global variables can be used throughout the program, and its scope is in the entire program. We can use global variables inside or outside the function.
A variable declared outside the function is the global variable by default. Python provides the **global** keyword to use global variable inside the function. If we don't use the **global** keyword, the function treats it as a local variable. Let's understand the following example.

Example -

```
# Declare a variable and initialize it
x = 101
 # Global variable in function
def mainFunction():
    # printing a global variable
    global x
    print(x)
    # modifying a global variable
    x = 'Welcome To Javatpoint'
    print(x)
  mainFunction()
print(x)
```

Output:

```
101
Welcome To Javatpoint
Welcome To Javatpoint
```

Chapter 3
Python Data Type

Python Data Types

Variables can hold values, and every value has a data-type. Python is a dynamically typed language; hence we do not need to define the type of the variable while declaring it. The interpreter implicitly binds the value with its type.

a = 5

The variable **a** holds integer value five and we did not define its type. Python interpreter will automatically interpret variables **a** as an integer type.

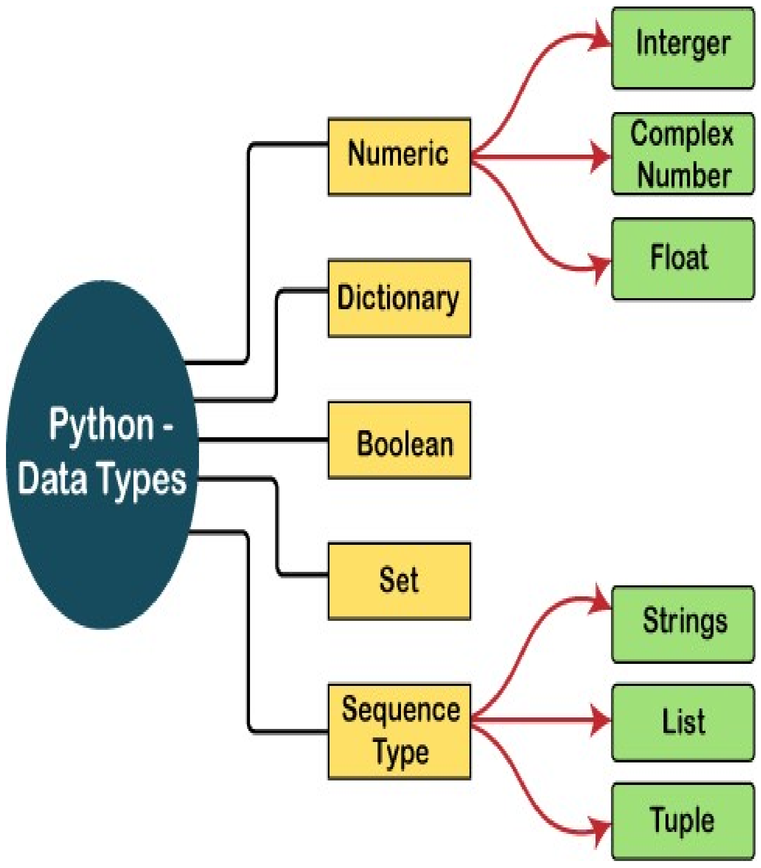
Python enables us to check the type of the variable used in the program. Python provides us the **type()** function, which returns the type of the variable passed.

Standard data types

A variable can hold different types of values. For example, a person's name must be stored as a string whereas its id must be stored as an integer.

Python provides various standard data types that define the storage method on each of them. The data types defined in Python are given below.

1. Numbers
2. Sequence Type
3. Boolean
4. Set
5. Dictionary



Numbers

Number stores numeric values. The integer, float, and complex values belong to a Python Numbers data-type. Python provides the **type()** function to know the data-type of the variable. Similarly, the **isinstance()** function is used to check an object belongs to a particular class.

Python supports three types of numeric data.

Int - Integer value can be any length such as integers 10, 2, 29, -20, -150 etc. Python has no restriction on the length of an integer. Its value belongs to **int**

Float - Float is used to store floating-point numbers like 1.9, 9.902, 15.2, etc. It is accurate upto 15 decimal points.

Complex - A complex number contains an ordered pair, i.e., x + iy where x and y denote the real and imaginary parts, respectively. The complex numbers like 2.14j, 2.0 + 2.3j, etc.

Sequence Type

3.3.1 String

The string can be defined as the sequence of characters represented in the quotation marks. In Python, we can use single, double, or triple quotes to define a string.
String handling in Python is a straightforward task since Python provides built-in functions and operators to perform operations in the string.
In the case of string handling, the operator + is used to concatenate two strings as the operation *"hello"+" python"* returns *"hello python"*.
The operator * is known as a repetition operator as the operation "Python" *2 returns 'Python Python'.

3.3.1 List

Python Lists are similar to arrays in C. However, the list can contain data of different types. The items stored in the list are separated with a comma (,) and enclosed within square brackets [].
We can use slice [:] operators to access the data of the list. The concatenation operator (+) and repetition operator (*) works with the list in the same way as they were working with the strings.

3.3.3 Tuple

A tuple is similar to the list in many ways. Like lists, tuples also contain the collection of the items of different data types. The items of the tuple are separated with a comma (,) and enclosed in parentheses ().
A tuple is a read-only data structure as we can't modify the size and value of the items of a tuple.

Dictionary

Dictionary is an unordered set of a key-value pair of items. It is like an associative array or a hash table where each key stores a specific value. Key can hold any primitive data type, whereas value is an arbitrary Python object.The items in the dictionary are separated with the comma (,) and enclosed in the curly braces {}.

Set

Python Set is the unordered collection of the data type. It is iterable, mutable(can modify after creation), and has unique elements. In set, the order of the elements is undefined; it may return the changed sequence of the element. The set is created by using a built-in function set(), or a sequence of elements is passed in the curly braces and separated by the comma. It can contain various types of values.

Boolean

Boolean type provides two built-in values, True and False. These values are used to determine the given statement true or false. It denotes by the class bool. True can be represented by any non-zero value or 'T' whereas false can be represented by the 0 or 'F'.

Chapter 4
Python Operator

Python Operator

The operator can be defined as a symbol which is responsible for a particular operation between two operands. Operators are the pillars of a program on which the logic is built in a specific programming language. Python provides a variety of operators, which are described as follows.

- Arithmetic operators
- Comparison operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations between two operands. It includes + (addition), - (subtraction), *(multiplication), /(divide), %(reminder), //(floor division), and exponent (**) operators.

Consider the following table for a detailed explanation of arithmetic operators.

Operator	Description
+ (Addition)	It is used to add two operands. For example, if a = 20, b = 10 => a+b = 30
- (Subtraction)	It is used to subtract the second operand from the first operand. If the first operand is less than the second operand, the value results negative. For example, if a = 20, b = 10 => a - b = 10
/ (divide)	It returns the quotient after dividing the first operand by the second operand. For example, if a = 20, b = 10 => a/b = 2.0
* (Multiplication)	It is used to multiply one operand with the other. For example, if a = 20, b = 10 => a * b = 200
% (reminder)	It returns the reminder after dividing the first operand by the second operand. For example, if a = 20, b = 10 => a%b = 0
** (Exponent)	It is an exponent operator represented as it calculates the first operand power to the second operand.
// (Floor division)	It gives the floor value of the quotient produced by dividing the two operands.

Comparison operator

Comparison operators are used to comparing the value of the two operands and returns Boolean true or false accordingly. The comparison operators are described in the following table.

Operator	Description
==	If the value of two operands is equal then the condition becomes true.
!=	If the value of two operands is not equal then the condition becomes true.
<=	If the first operand is less than or equal to the second operand then the condition becomes true.
>=	If the first operand is greater than or equal to the second operand then the condition becomes true.
>	If the first operand is greater than the second operand then the condition becomes true.
<	If the first operand is less than the second operand then the condition becomes true.

Assignment Operators

The assignment operators are used to assign the value of the right expression to the left operand. The assignment operators are described in the following table.

Operator	Description
=	It assigns the value of the right expression to the left operand.
+=	It increases the value of the left operand by the value of the right operand and assigns the modified value back to left operand. For example, if a = 10, b = 20 => a+ = b will be equal to a = a+ b and therefore, a = 30.
-=	It decreases the value of the left operand by the value of the right operand and assigns the modified value back to left operand. For example, if a = 20, b = 10 => a- = b will be equal to a = a- b and therefore, a = 10.
*=	It multiplies the value of the left operand by the value of the right operand and assigns the modified value back to then the left operand. For example, if a = 10, b = 20 => a* = b will be equal to a = a* b and therefore, a = 200.
%=	It divides the value of the left operand by the value of the right operand and assigns the reminder back to the left operand. For example, if a = 20, b = 10 => a % = b will be equal to a = a % b and therefore, a = 0.
**=	a**=b will be equal to a=a**b, for example, if a = 4, b =2, a**=b will assign 4**2 = 16 to a.
//=	A//=b will be equal to a = a// b, for example, if a = 4, b = 3, a//=b will assign 4//3 = 1 to a.

Logical Operators

The logical operators are used primarily in the expression evaluation to make a decision. Python supports the following logical operators.

Operator	Description
and	If both the expression are true, then the condition will be true. If a and b are the two expressions, a → true, b → true => a and b → true.
Or	If one of the expressions is true, then the condition will be true. If a and b are the two expressions, a → true, b → false => a or b → true.
not	If an expression a is true, then not (a) will be false and vice versa.

Bitwise Operators

The bitwise operators perform bit by bit operation on the values of the two operands. Consider the following example.

For example,
if a = 7
 b = 6
then, binary (a) = 0111
 binary (b) = 0011
hence, a & b = 0011
 a | b = 0111
 a ^ b = 0100
 ~ a = 1000

Operator	Description
& (binary and)	If both the bits at the same place in two operands are 1, then 1 is copied to the result. Otherwise, 0 is copied.
| (binary or)	The resulting bit will be 0 if both the bits are zero; otherwise, the resulting bit will be 1.
^ (binary xor)	The resulting bit will be 1 if both the bits are different; otherwise, the resulting bit will be 0.
~ (negation)	It calculates the negation of each bit of the operand, i.e., if the bit is 0, the resulting bit will be 1 and vice versa.
<< (left shift)	The left operand value is moved left by the number of bits present in the right operand.
>> (right shift)	The left operand is moved right by the number of bits present in the right operand.

Membership Operators

Python membership operators are used to check the membership of value inside a Python data structure. If the value is present in the data structure, then the resulting value is true otherwise it returns false.

Operator	Description
in	It is evaluated to be true if the first operand is found in the second operand (list, tuple, or dictionary).
not in	It is evaluated to be true if the first operand is not found in the second operand (list, tuple, or dictionary).

Identity Operators

The identity operators are used to decide whether an element certain class or type.

Operator	Description
Is	It is evaluated to be true if the reference present at both sides point to the same object.
is not	It is evaluated to be true if the reference present at both sides do not point to the same object.

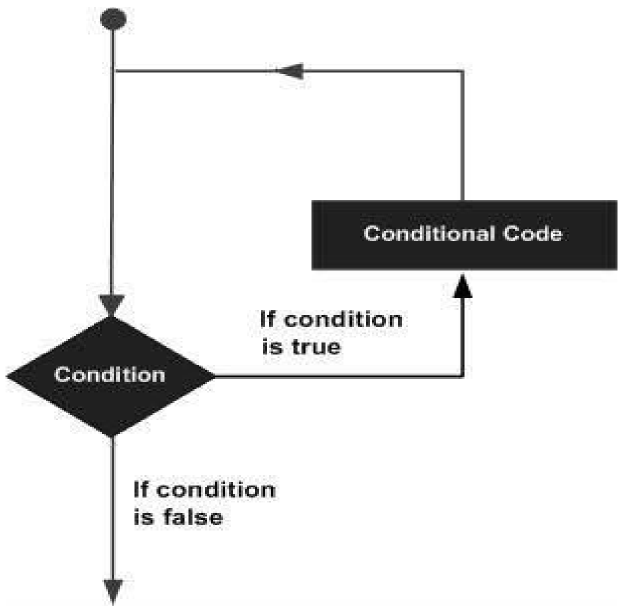
Chapter 5
Python Loops

Python Loops

In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on. There may be a situation when you need to execute a block of code several number of times.

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times. The following diagram illustrates a loop statement –



While Loop

A **while** loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

Syntax

The syntax of a **while** loop in Python programming language is −

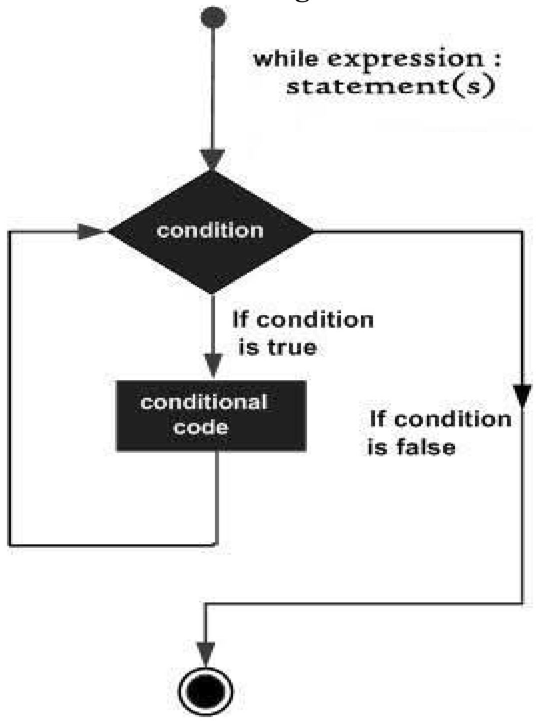
```
while expression:
   statement(s)
```

Here, **statement(s)** may be a single statement or a block of statements. The **condition** may be any expression, and true is any non-zero value. The loop iterates while the condition is true.

When the condition becomes false, program control passes to the line immediately following the loop.

In Python, all the statements indented by the same number of character spaces after a programming construct are considered to be part of a single block of code. Python uses indentation as its method of grouping statements.

Flow Diagram



Here, key point of the while loop is that the loop might not ever run. When the condition is tested and the result is false, the loop body will be skipped and the first statement after the while loop will be executed.

The Infinite Loop

A loop becomes infinite loop if a condition never becomes FALSE. You must use caution when using while loops because of the possibility that this condition never resolves to a FALSE value. This results in a loop that never ends. Such a loop is called an infinite loop.
An infinite loop might be useful in client/server programming where the server needs to run continuously so that client programs can communicate with it as and when required.

Using else Statement with While Loop

Python supports to have an else statement associated with a loop statement.

- If the else statement is used with a while loop, the else statement is executed when the condition becomes false.

Single Statement Suites

Similar to the **if** statement syntax, if your **while** clause consists only of a single statement, it may be placed on the same line as the while header.
Here is the syntax and example of a **one-line while** clause −

```
#!/usr/bin/python
flag = 1
while (flag): print 'Given flag is really true!'
print "Good bye!"
```

For Loop

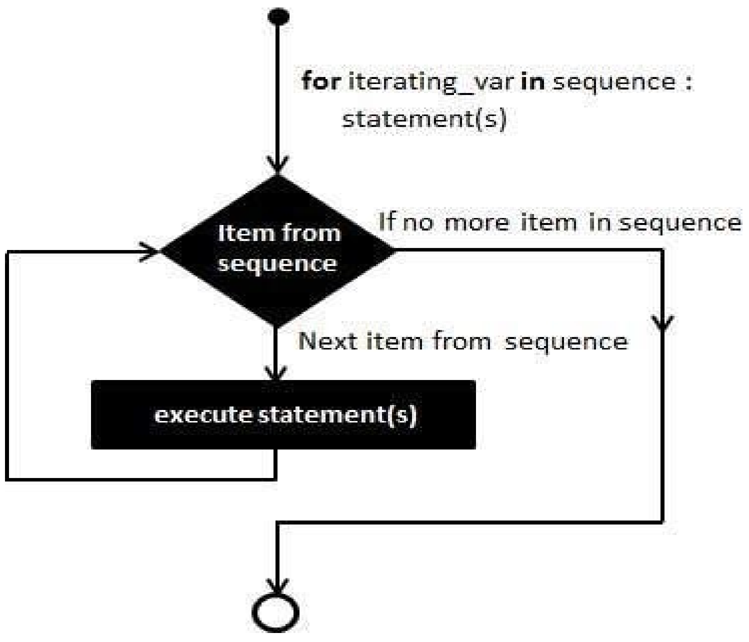
It has the ability to iterate over the items of any sequence, such as a list or a string.

Syntax

```
for iterating_var in sequence:
   statements(s)
```

If a sequence contains an expression list, it is evaluated first. Then, the first item in the sequence is assigned to the iterating variable *iterating_var*. Next, the statements block is executed. Each item in the list is assigned to *iterating_var*, and the statement(s) block is executed until the entire sequence is exhausted.

Flow Diagram



Iterating by Sequence Index

An alternative way of iterating through each item is by index offset into the sequence itself.

Following is a simple example −

```
#!/usr/bin/python
fruits = ['banana', 'apple', 'mango']
for index in range(len(fruits)):
print 'Current fruit :', fruits[index]
print "Good bye!"
```

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

```
Current fruit : banana
Current fruit : apple
Current fruit : mango
Good bye!
```

Here, we took the assistance of the len() built-in function, which provides the total number of elements in the tuple as well as the range() built-in function to give us the actual sequence to iterate over.

Using else Statement with For Loop

Python supports to have an else statement associated with a loop statement

If the else statement is used with a for loop, the else statement is executed when the loop has exhausted iterating the list.

Nested Loop

Python programming language allows to use one loop inside another loop. Following section shows few examples to illustrate the concept.

Syntax

```
for iterating_var in sequence:
   for iterating_var in sequence:
      statements(s)
   statements(s)
```

The syntax for a **nested while loop** statement in Python programming language is as follows −

```
while expression:
   while expression:
      statement(s)
   statement(s)
```

A final note on loop nesting is that you can put any type of loop inside of any other type of loop.

Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

Break Statement

It terminates the current loop and resumes execution at the next statement, just like the traditional break statement in C.

The most common use for break is when some external condition is triggered requiring a hasty exit from a loop. The **break** statement can be used in both *while* and *for* loops.

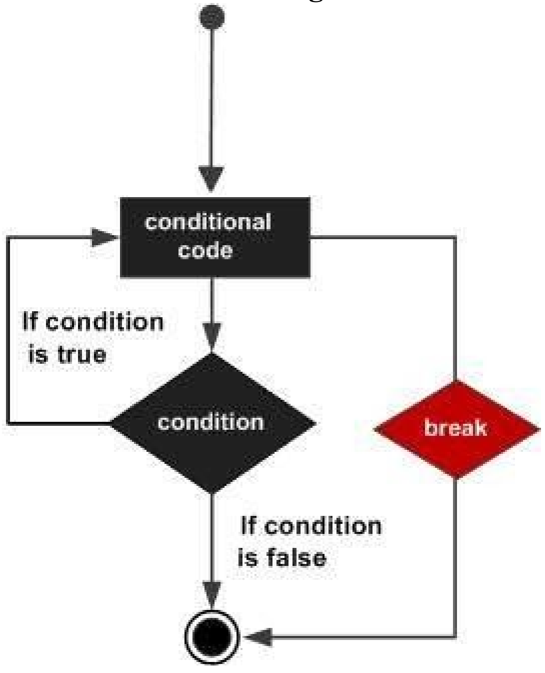
If you are using nested loops, the break statement stops the execution of the innermost loop and start executing the next line of code after the block.

Syntax

The syntax for a **break** statement in Python is as follows −

```
break
```

Flow Diagram



Continue Statement

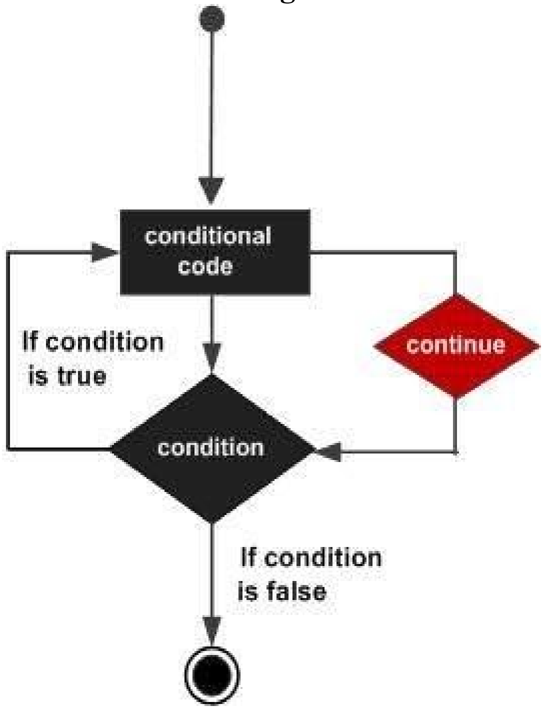
It returns the control to the beginning of the while loop.. The **continue** statement rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop.

The **continue** statement can be used in both *while* and *for* loops.

Syntax

continue

Flow Diagram



Pass Statement

It is used when a statement is required syntactically but you do not want any command or code to execute.

The pass statement is a *null* operation; nothing happens when it executes. The pass is also useful in places where your code will eventually go, but has not been written yet (e.g., in stubs for example) –

Syntax

pass

Chapter 6
Python Function

Python Function

Functions are the most important aspect of an application. A function can be defined as the organized block of reusable code, which can be called whenever required.
Python allows us to divide a large program into the basic building blocks known as a function. The function contains the set of programming statements enclosed by {}. A function can be called multiple times to provide reusability and modularity to the Python program.
The Function helps to programmer to break the program into the smaller part. It organizes the code very effectively and avoids the repetition of the code. As the program grows, function makes the program more organized.
Python provide us various inbuilt functions like **range()** or **print()**. Although, the user can create its functions, which can be called user-defined functions.
There are mainly two types of functions.

- **User-define functions** - The user-defined functions are those define by the **user** to perform the specific task.
- **Built-in functions** - The built-in functions are those functions that are **pre-defined** in Python.

Advantage of Functions in Python

There are the following advantages of Python functions.

- Using functions, we can avoid rewriting the same logic/code again and again in a program.
- We can call Python functions multiple times in a program and anywhere in a program.
- We can track a large Python program easily when it is divided into multiple functions.
- Reusability is the main achievement of Python functions.
- However, Function calling is always overhead in a Python program.

Creating a Function

Python provides the **def** keyword to define the function. The syntax of the define function is given below.
Syntax:

```
def my_function(parameters):
    function_block
return expression
```

Let's understand the syntax of functions definition.

- The **def** keyword, along with the function name is used to define the function.
- The identifier rule must follow the function name.
- A function accepts the parameter (argument), and they can be optional.
- The function block is started with the colon (:), and block statements must be at the same indentation.
- The **return** statement is used to return the value. A function can have only one **return**

Function Calling

In Python, after the function is created, we can call it from another function. A function must be defined before the function call; otherwise, the Python interpreter gives an error. To call the function, use the function name followed by the parentheses.
Consider the following example of a simple example that prints the message "Hello World".

```
#function definition
def hello_world():
   print("hello world")
# function calling
```

```
hello_world()
```
Output:
```
hello world
```

The return statement
The return statement is used at the end of the function and returns the result of the function. It terminates the function execution and transfers the result where the function is called. The return statement cannot be used outside of the function.

Syntax

return [expression_list]

It can contain the expression which gets evaluated and value is returned to the caller function. If the return statement has no expression or does not exist itself in the function then it returns the **None** object.

Arguments in function
The arguments are types of information which can be passed into the function. The arguments are specified in the parentheses. We can pass any number of arguments, but they must be separate them with a comma.

Example 1
```
#defining the function
def func (name):
    print("Hi ",name)
#calling the function
func("Devansh")
```
Output:
```
Hi Devansh
```

Call by reference in Python

In Python, call by reference means passing the actual value as an argument in the function. All the functions are called by reference, i.e., all the changes made to the reference inside the function revert back to the original value referred by the reference.

Types of arguments

There may be several types of arguments which can be passed at the time of function call.

1. Required arguments
2. Keyword arguments
3. Default arguments
4. Variable-length arguments

1 Required Arguments

Till now, we have learned about function calling in Python. However, we can provide the arguments at the time of the function call. As far as the required arguments are concerned, these are the arguments which are required to be passed at the time of function calling with the exact match of their positions in the function call and function definition. If either of the arguments is not provided in the function call, or the position of the arguments is changed, the Python interpreter will show the error.

2 Keyword arguments(**kwargs)

Python allows us to call the function with the keyword arguments. This kind of function call will enable us to pass the arguments in the random order.

The name of the arguments is treated as the keywords and matched in the function calling and definition. If the same match is found, the values of the arguments are copied in the function definition.

3 Default Arguments

Python allows us to initialize the arguments at the function definition. If the value of any of the arguments is not provided at the time of function call, then that argument can be initialized with the value given in the definition even if the argument is not specified at the function call.

4 Variable-length Arguments (*args)

In large projects, sometimes we may not know the number of arguments to be passed in advance. In such cases, Python provides us the flexibility to offer the comma-separated values which are internally treated as tuples at the function call. By using the variable-length arguments, we can pass any number of arguments.

Chapter 7
Python Modules

Python Modules

A python module can be defined as a python program file which contains a python code including python functions, class, or variables. In other words, we can say that our python code file saved with the extension (.py) is treated as the module. We may have a runnable code inside the python module.

Modules in Python provides us the flexibility to organize the code in a logical way.

To use the functionality of one module into another, we must have to import the specific module.

Example

In this example, we will create a module named as file.py which contains a function func that contains a code to print some message on the console.

Let's create the module named as **file.py.**

```
#displayMsg prints a message to the name being passed.
def displayMsg(name)
   print("Hi "+name);
```

Here, we need to include this module into our main module to call the method displayMsg() defined in the module named file.

Loading the module in our python code

We need to load the module in our python code to use its functionality. Python provides two types of statements as defined below.

1. The import statement
2. The from-import statement

The import statement

The import statement is used to import all the functionality of one module into another. Here, we must notice that we can use the functionality of any python source file by importing that file as the module into another python source file.

We can import multiple modules with a single import statement, but a module is loaded once regardless of the number of times, it has been imported into our file.

The syntax to use the import statement is given below.

```
import module1,module2,........ module n
```

Hence, if we need to call the function displayMsg() defined in the file file.py, we have to import that file as a module into our module as shown in the example below.

Example:

```
import file;
name = input("Enter the name?")
file.displayMsg(name)
```

Output:

```
Enter the name?John
   Hi John
```

The from-import statement

Instead of importing the whole module into the namespace, python provides the flexibility to import only the specific attributes of a module. This can be done by using from? import statement. The syntax to use the from-import statement is given below.

1. **from** < module-name> **import** <name 1>, <name 2>..,<name n>

Consider the following module named as calculation which contains three functions as summation, multiplication, and divide.

calculation.py:

```
#place the code in the calculation.py
def summation(a,b):
    return a+b
def multiplication(a,b):
    return a*b;
def divide(a,b):
    return a/b;
```

Main.py:

```
from calculation import summation
#it will import only the summation() from calculation.py
a = int(input("Enter the first number"))
b = int(input("Enter the second number"))
print("Sum = ",summation(a,b)) #we do not need to specify the module name while accessing s
ummation()
```

Output:

```
Enter the first number10
Enter the second number20
Sum =  30
```

The from...import statement is always better to use if we know the attributes to be imported from the module in advance. It doesn't let our code to be heavier. We can also import all the attributes from a module by using *.

Consider the following syntax.

from <module> **import** *

Renaming A Module

Python provides us the flexibility to import some module with a specific name so that we can use this name to use that module in our python source file.

The syntax to rename a module is given below.

1. **import** <module-name> as <specific-name>

1. Example

```
1.  #the module calculation of previous example is imported in this example as cal.
2.  import calculation as cal;
3.  a = int(input("Enter a?"));
4.  b = int(input("Enter b?"));
5.  print("Sum = ",cal.summation(a,b))
```

Output:

```
Enter a?10
Enter b?20
Sum =  30
```

Using dir() function

The dir() function returns a sorted list of names defined in the passed module. This list contains all the sub-modules, variables and functions defined in this module.

Consider the following example.

Example

```
import json

List = dir(json)

print(List)
```

Output:

```
['JSONDecoder', 'JSONEncoder', '__all__', '__author__', '__builtins__', '__cached__', '__doc__',
'_file_', '_loader_', '_name_', '_package_', '_path_', '_spec_', '_version_',
'_default_decoder', '_default_encoder', 'decoder', 'dump', 'dumps', 'encoder', 'load', 'loads', 'scanner']
```

The reload() function

As we have already stated that, a module is loaded once regardless of the number of times it is imported into the python source file. However, if you want to reload the already imported module to re-execute the top-level code, python provides us the reload() function. The syntax to use the reload() function is given below.

```
reload(<module-name>)
```

for example, to reload the module calculation defined in the previous example, we must use the following line of code.

```
reload(calculation)
```

Scope of variables

In Python, variables are associated with two types of scopes. All the variables defined in a module contain the global scope unless or until it is defined within a function.
All the variables defined inside a function contain a local scope that is limited to this function itself. We can not access a local variable globally.
If two variables are defined with the same name with the two different scopes, i.e., local and global, then the priority will always be given to the local variable.
Consider the following example.

Example

```
name = "john"
def print_name(name):
   print("Hi",name) #prints the name that is local to this function only.
name = input("Enter the name?")
print_name(name)
```

Output:

```
Hi David
```

Python PIP

What is PIP?
PIP is a package manager for Python packages, or modules if you like.
Note: If you have Python version 3.4 or later, PIP is included by default.

What is a Package?
A package contains all the files you need for a module.
Modules are Python code libraries you can include in your project.

Check if PIP is Installed
Navigate your command line to the location of Python's script directory, and type the following:
Example
Check PIP version:

```
C:\Users\Your Name\AppData\Local\Programs\Python\Python36-32\Scripts>pip --version
```

Install PIP
If you do not have PIP installed, you can download and install it from this page: https://pypi.org/project/pip/

Download a Package
Downloading a package is very easy.
Open the command line interface and tell PIP to download the package you want.
Navigate your command line to the location of Python's script directory, and type the following:
Example
Download a package named "camelcase":

```
C:\Users\Your Name\AppData\Local\Programs\Python\Python36-32\Scripts>pip install camelcase
```

Now you have downloaded and installed your first package!

Using a Package
Once the package is installed, it is ready to use.
Import the "camelcase" package into your project.

Remove a Package
Use the uninstall command to remove a package:
Example
Uninstall the package named "camelcase":

```
C:\Users\Your Name\AppData\Local\Programs\Python\Python36-32\Scripts>pip uninstall camelcase
```

The PIP Package Manager will ask you to confirm that you want to remove the camelcase package:

```
Uninstalling camelcase-02.1: Would remove:
 c:\users\Your Name\appdata\local\programs\python\python36-32\lib\site-packages\camecase-0.2-py3.6.egg-info
c:\users\Your Name\appdata\local\programs\python\python36-32\lib\site-packages\camecase\*
Proceed (y/n)?
```

Press y and the package will be removed.

List Packages
Use the list command to list all the packages installed on your system:
Example
List installed packages:

```
C:\Users\Your Name\AppData\Local\Programs\Python\Python36-32\Scripts>pip list
```

Result:

```
Package         Version
-----------------------
camelcase       0.2
mysql-connector 2.1.6
pip             18.1
pymongo         3.6.1
setuptools      39.0.1
```

Chapter 8
Python OOPs Concepts

Python OOPs Concepts

Like other general-purpose programming languages, Python is also an object-oriented language since its beginning. It allows us to develop applications using an Object-Oriented approach. In Python, we can easily create and use classes and objects.
An object-oriented paradigm is to design the program using classes and objects. The object is related to real-word entities such as book, house, pencil, etc. The oops concept focuses on writing the reusable code. It is a widespread technique to solve the problem by creating objects.
Major principles of object-oriented programming system are given below.

- Class
- Object
- Method
- Inheritance
- Polymorphism
- Data Abstraction
- Encapsulation

Class

The class can be defined as a collection of objects. It is a logical entity that has some specific attributes and methods. For example: if you have an employee class, then it should contain an attribute and method, i.e. an email id, name, age, salary, etc.
Syntax

```
class ClassName:
        <statement-1>
        .
        .
        <statement-N>
```

Object

The object is an entity that has state and behavior. It may be any real-world object like the mouse, keyboard, chair, table, pen, etc.
Everything in Python is an object, and almost everything has attributes and methods. All functions have a built-in attribute_doc_, which returns the docstring defined in the function source code.
When we define a class, it needs to create an object to allocate the memory. Consider the following example.
Example:

```
class car:
    def_init_(self,modelname, year):
        self.modelname = modelname
        self.year = year
    def display(self):
        print(self.modelname,self.year)
c1 = car("Toyota", 2016)
c1.display()
```

Output:

```
Toyota 2016
```

In the above example, we have created the class named car, and it has two attributes modelname and year. We have created a c1 object to access the class attribute. The c1 object will allocate memory for these values. We will learn more about class and object in the next tutorial.

Method

The method is a function that is associated with an object. In Python, a method is not unique to class instances. Any object type can have methods.

Inheritance

Inheritance is the most important aspect of object-oriented programming, which simulates the real-world concept of inheritance. It specifies that the child object acquires all the properties and behaviors of the parent object.
By using inheritance, we can create a class which uses all the properties and behavior of another class. The new class is known as a derived class or child class, and the one whose properties are acquired is known as a base class or parent class.
It provides the re-usability of the code.

Polymorphism

Polymorphism contains two words "poly" and "morphs". Poly means many, and morph means shape. By polymorphism, we understand that one task can be performed in different ways. For example - you have a class animal, and all animals speak. But they speak differently. Here, the "speak" behavior is polymorphic in a sense and depends on the animal. So, the abstract "animal" concept does not actually "speak", but specific animals (like dogs and cats) have a concrete implementation of the action "speak".

Encapsulation

Encapsulation is also an essential aspect of object-oriented programming. It is used to restrict access to methods and variables. In encapsulation, code and data are wrapped together within a single unit from being modified by accident.

Data Abstraction

Data abstraction and encapsulation both are often used as synonyms. Both are nearly synonyms because data abstraction is achieved through encapsulation.
Abstraction is used to hide internal details and show only functionalities. Abstracting something means to give names to things so that the name captures the core of what a function or a whole program does.

Chapter 9
Framework

Framework

As a programmer, you don't need to start from scratch when you have tools designed to help you with your projects. Frameworks are software that is developed and used by developers to build applications.

What is Frameworks?

Since they are often built, tested, and optimized by several experienced software engineers and programmers, software frameworks are versatile, robust, and efficient.
Using a software framework to develop applications lets you focus on the high-level functionality of the application. This is because any low-level functionality is taken care of by the framework itself.

Why do we use Frameworks?

Developing software is a complex process. It necessitates a plethora of tasks, including coding, designing, and testing. For only the coding part, programmers had to take care of the syntax, declarations, garbage collection, statements, exceptions, and more.
Software frameworks make life easier for developers by allowing them to take control of the entire software development process, or most of it, from a single platform.

Advantages of using a software framework:

- Assists in establishing better programming practices and fitting use of design patterns
- Code is more secure
- Duplicate and redundant code can be avoided
- Helps consistent developing code with fewer bugs
- Makes it easier to work on sophisticated technologies
- One could create their software framework or contribute to open-source frameworks. Hence, there is a continuous improvement in the functionality
- Several code segments and functionalities are pre-built and pre-tested. This makes applications more reliable
- Testing and debugging the code is a lot easier and can be done even by developers who do not own the code
- The time required to develop an application is reduced significantly

Python Tkinter

Tkinter tutorial provides basic and advanced concepts of Python Tkinter. Our Tkinter tutorial is designed for beginners and professionals.
Python provides the standard library Tkinter for creating the graphical user interface for desktop based applications.
Developing desktop based applications with python Tkinter is not a complex task. An empty Tkinter top-level window can be created by using the following steps.

1. import the Tkinter module.
2. Create the main application window.
3. Add the widgets like labels, buttons, frames, etc. to the window.
4. Call the main event loop so that the actions can take place on the user's computer screen.

Tkinter Modules

Most of the time, tkinter is all you really need, but a number of additional modules are available as well. The Tk interface is located in a binary module named _tkinter. This module contains the low-level interface to Tk, and should never be used directly by application programmers. It is usually a shared library (or DLL), but might in some cases be statically linked with the Python interpreter.

In addition to the Tk interface module, tkinter includes a number of Python modules, tkinter.constants being one of the most important. Importing tkinter will automatically import tkinter.constants, so, usually, to use Tkinter all you need is a simple import statement:
import tkinter
Or, more often:
from tkinter import *
class tkinter.**Tk**(*screenName=None*, *baseName=None*, *className='Tk'*, *useTk=1*)
The Tk class is instantiated without arguments. This creates a toplevel widget of Tk which usually is the main window of an application. Each instance has its own associated Tcl interpreter.
tkinter.**Tcl**(*screenName=None*, *baseName=None*, *className='Tk'*, *useTk=0*)
The Tcl() function is a factory function which creates an object much like that created by the Tk class, except that it does not initialize the Tk subsystem. This is most often useful when driving the Tcl interpreter in an environment where one doesn't want to create extraneous toplevel windows, or where one cannot (such as Unix/Linux systems without an X server). An object created by the Tcl() object can have a Toplevel window created (and the Tk subsystem initialized) by calling its loadtk() method.
Other modules that provide Tk support include:
tkinter.colorchooser
Dialog to let the user choose a color.
tkinter.commondialog
Base class for the dialogs defined in the other modules listed here.
tkinter.filedialog
Common dialogs to allow the user to specify a file to open or save.
tkinter.font
Utilities to help work with fonts.
tkinter.messagebox
Access to standard Tk dialog boxes.
tkinter.scrolledtext
Text widget with a vertical scroll bar built in.
tkinter.simpledialog
Basic dialogs and convenience functions.
tkinter.dnd
Drag-and-drop support for tkinter. This is experimental and should become deprecated when it is replaced with the Tk DND.
turtle
Turtle graphics in a Tk window.

Tkinker Widgets

There are various widgets like button, canvas, checkbutton, entry, etc. that are used to build the python GUI applications.

SN	Widget	Description
1	Button	The Button is used to add various kinds of buttons to the python application.
2	Canvas	The canvas widget is used to draw the canvas on the window.
3	Checkbutton	The Checkbutton is used to display the CheckButton on the window.
4	Entry	The entry widget is used to display the single-line text field to the user. It is commonly used to accept user values.
5	Frame	It can be defined as a container to which, another widget can be added and organized.
6	Label	A label is a text used to display some message or information about the other widgets.

7	ListBox	The ListBox widget is used to display a list of options to the user.
8	Menubutton	The Menubutton is used to display the menu items to the user.
9	Menu	It is used to add menu items to the user.
10	Message	The Message widget is used to display the message-box to the user.
11	Radiobutton	The Radiobutton is different from a checkbutton. Here, the user is provided with various options and the user can select only one option among them.
12	Scale	It is used to provide the slider to the user.
13	Scrollbar	It provides the scrollbar to the user so that the user can scroll the window up and down.
14	Text	It is different from Entry because it provides a multi-line text field to the user so that the user can write the text and edit the text inside it.
14	Toplevel	It is used to create a separate window container.
15	Spinbox	It is an entry widget used to select from options of values.
16	PanedWindow	It is like a container widget that contains horizontal or vertical panes.
17	LabelFrame	A LabelFrame is a container widget that acts as the container
18	MessageBox	This module is used to display the message-box in the desktop based applications.

Python Tkinter Geometry

The Tkinter geometry specifies the method by using which, the widgets are represented on display. The python Tkinter provides the following geometry methods.

1. The pack() method
2. The grid() method
3. The place() method

Let's discuss each one of them in detail.

Python Tkinter pack() method

The pack() widget is used to organize widget in the block. The positions widgets added to the python application using the pack() method can be controlled by using the various options specified in the method call.
However, the controls are less and widgets are generally added in the less organized manner.
The syntax to use the pack() is given below.
syntax
widget.pack(options)

A list of possible options that can be passed in pack() is given below.

- expand: If the expand is set to true, the widget expands to fill any space.
- Fill: By default, the fill is set to NONE. However, we can set it to X or Y to determine whether the widget contains any extra space.
- size: it represents the side of the parent to which the widget is to be placed on the window.

Python Tkinter grid() method

The grid() geometry manager organizes the widgets in the tabular form. We can specify the rows and columns as the options in the method call. We can also specify the column span (width) or rowspan(height) of a widget.
This is a more organized way to place the widgets to the python application. The syntax to use the grid() is given below.

Syntax
widget.grid(options)

A list of possible options that can be passed inside the grid() method is given below.

- **Column**
  The column number in which the widget is to be placed. The leftmost column is represented by 0.
- **Columnspan**
  The width of the widget. It represents the number of columns up to which, the column is expanded.
- **ipadx, ipady**
  It represents the number of pixels to pad the widget inside the widget's border.
- **padx, pady**
  It represents the number of pixels to pad the widget outside the widget's border.
- **row**
  The row number in which the widget is to be placed. The topmost row is represented by 0.
- **rowspan**
  The height of the widget, i.e. the number of the row up to which the widget is expanded.
- **Sticky**
  If the cell is larger than a widget, then sticky is used to specify the position of the widget inside the cell. It may be the concatenation of the sticky letters representing the position of the widget. It may be N, E, W, S, NE, NW, NS, EW, ES.

Python Tkinter place() method
The place() geometry manager organizes the widgets to the specific x and y coordinates.
Syntax
widget.place(options)

A list of possible options is given below.

- **Anchor:** It represents the exact position of the widget within the container. The default value (direction) is NW (the upper left corner)
- **bordermode:** The default value of the border type is INSIDE that refers to ignore the parent's inside the border. The other option is OUTSIDE.
- **height, width:** It refers to the height and width in pixels.
- **relheight, relwidth:** It is represented as the float between 0.0 and 1.0 indicating the fraction of the parent's height and width.
- **relx, rely:** It is represented as the float between 0.0 and 1.0 that is the offset in the horizontal and vertical direction.
- **x, y:** It refers to the horizontal and vertical offset in the pixels.

Chapter 10
Project
(YOUTUBE VIDEO DOWNLODER APPLICATION)

Introduction

Python YouTube Video Downloader is an application to download videos and playlist from YouTube. This provides users to download videos they need in their devices and watch them offline
The Youtube downloader project is a python project. The object of this project is to download any type of video in a fast and easy way from youtube in your device.
In this python project, user has to copy the youtube video URL that they want to download and simply paste that URL in the 'paste link here' section, And select the download file location, And click on the download button, it will start downloading the video. When video downloading finishes, it shows a message 'downloaded' popup on the window below the download button. Any reason viode is not download

Dependencies

- Import os
- Import shutil
- Import tkinter
- Import ttk
- From tkinter.filedialog, messagebox
- Import askdirectory
- Import PIL
- Import Pytube

Important Required Modules

Os
Utility module for OS interaction in programs The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc.
There are some functions in the OS module which are given below:

os.name()
This function provides the name of the operating system module that it imports.

os.mkdir()
The os.mkdir() function is used to create new directory. Consider the following example.
import os
os.mkdir("d:\\newdir")
It will create the new directory to the path in the string argument of the function in the D drive named folder newdir.

os.getcwd()
It returns the current working directory(CWD) of the file.

os.chdir()
The os module provides the chdir() function to change the current working directory.

os.rmdir()
The **rmdir()** function removes the specified directory with an absolute or related path. First, we have to change the current working directory and remove the folder.

os.error()
The os.error() function defines the OS level errors. It raises OSError in case of invalid or inaccessible file names and path etc.

os.popen()
This function opens a file or from the command specified, and it returns a file object which is connected to a pipe.

os.close()
This function closes the associated file with descriptor **fr**.

os.rename()
A file or directory can be renamed by using the function **os.rename()**. A user can rename the file if it has privilege to change the file.

os.access()
This function uses real **uid/gid** to test if the invoking user has access to the path.

<u>**Shutil**</u>

Shutil module in Python provides many functions of high-level operations on files and collections of files. This module helps in automating process of copying and removal of files and directories.

shutil.copyfile() method in Python is used to copy the content of source file to destination file. Metadata of the file is not copied. Source and destination must represent a file and destination must be writable. If destination already exists then it will be replaced with the source file otherwise a new file will be created.
If source and destination represents the same file then SameFileError exception will be raised.

Syntax: shutil.copyfile(source, destination, *, follow_symlinks = True)
source: A string representing the path of the source file.
destination: A string representing the path of the destination file.
follow_symlinks (optional) : The default value of this parameter is True. If False and source represents a symbolic link then a new symbolic link will be created instead of copying the file.
Note: The '*' in parameter list indicates that all following parameters (Here in our case 'follow_symlinks') are keyword-only parameters and they can be provided using their name, not as positional parameter.

Tkinter

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.
More information for Tkinter page no.31

Filedialog

The tkinter.filedialog module provides classes and factory functions for creating file/directory selection windows.

Easygui

This library is uesd for folder selection.

Imageio

Used to read the file which is chosen by file box using a path.

Messagebox

The tkMessageBox module is used to display message boxes in your applications. This module provides a number of functions that you can use to display an appropriate message.

Pillow (PIL)

The Python Imaging Library adds image processing capabilities to your Python interpreter.
This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.
The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

pytube

pytube is a very serious, lightweight, dependency-free Python library (and command-line utility) for downloading YouTube Videos.

Installation

To install from pypi with pip:
```
$ python -m pip install pytube
```
Sometime, the pypi release becomes slightly outdated. To install from the source with pip:
```
$ python -m pip install git+https://github.com/nficano/pytube
```

Description

YouTube is the most popular video-sharing platform in the world and as a hacker you may encounter a situation where you want to script something to download videos. For this I present to you *pytube*.
pytube is a lightweight library written in Python. It has no third party dependencies and aims to be highly reliable.
pytube also makes pipelining easy, allowing you to specify callback functions for different download events, such as on progress or on complete.

Hardware Requirement

- **Processor**
  Pentium 2.0 and above.
- **Ram**
  Minimum 2 Gb .
- **Hard Disk**
  250 Gb.
- **Any Pointer Device**
  Mouse

Software Requirement

- **Operating System**
  Windows XP , Windows7, 8 & 10 etc.
- **Code editor**
  Visual Studio code & PyCharm.
- **Language**
  Python.
- **Framework**
  Python Tkiner.

Features

- Easy to use not so complex GUI interface.
- Easy to separate all file.
- User friendly.
- This application is secure because this application work offline own your system and not share your information.
- This application is save your lot of time youtube video & playlist download.
- It is use in both places professional and non-professional.

<u>PROJECT LINK</u>

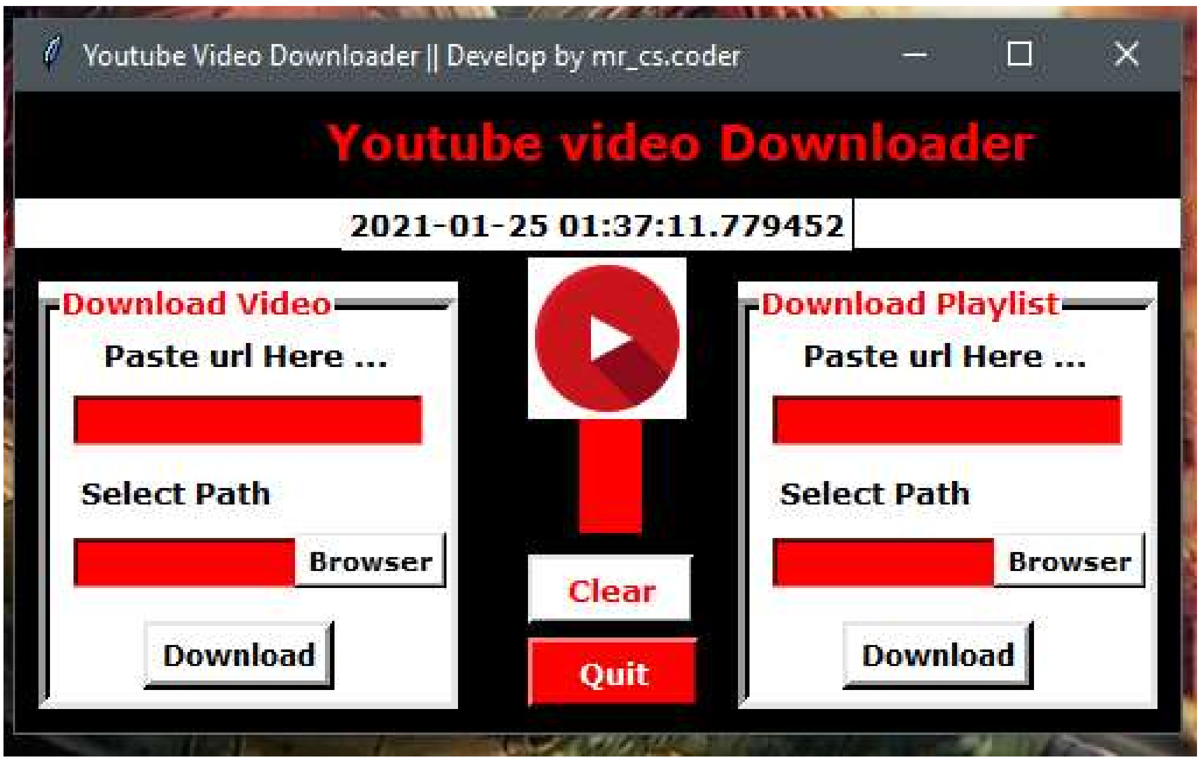
⇩

https://github.com/Akshat303/YouTube Video Downloder Application

Screenshorts

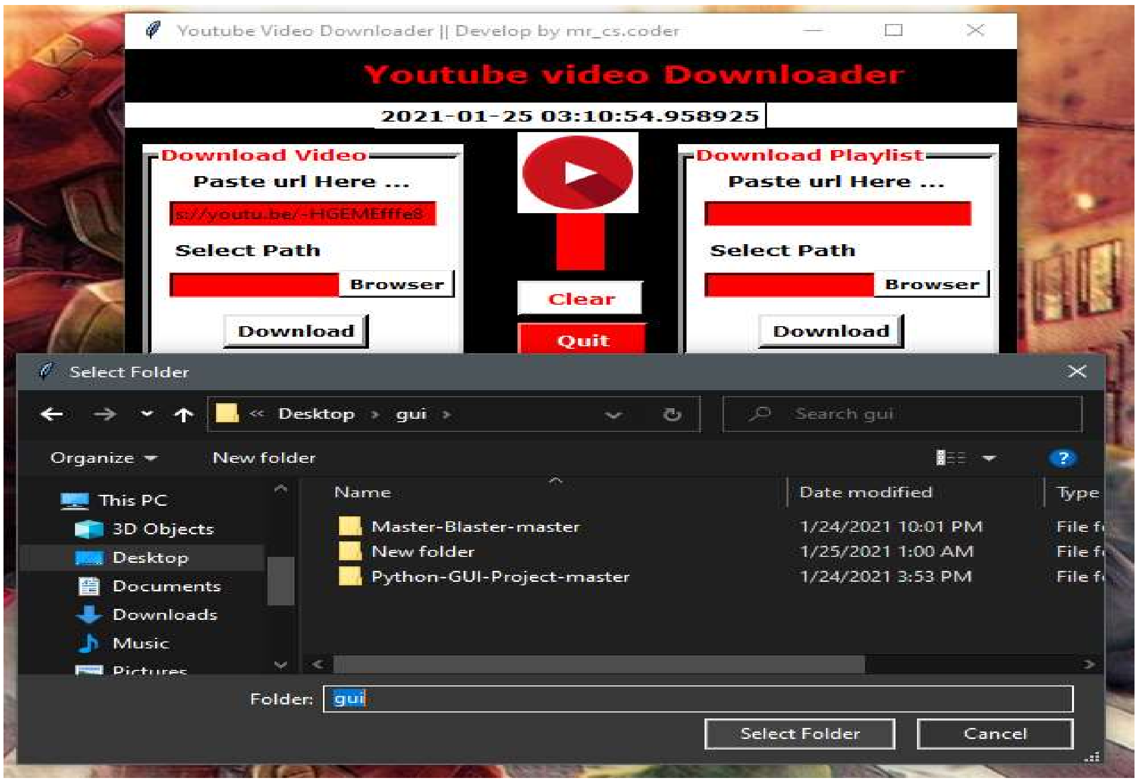
Project GUI Interface

⇩



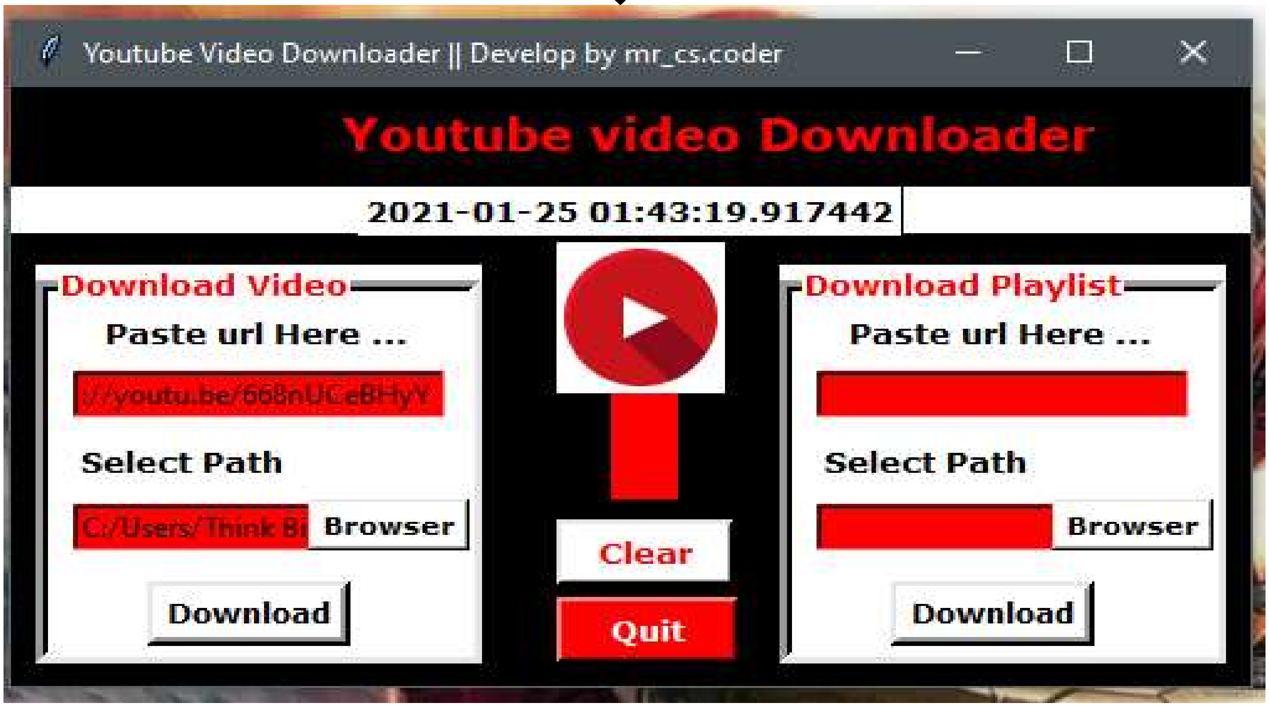
Paste Video / Playlist Url And Sellect Download Location

⇓



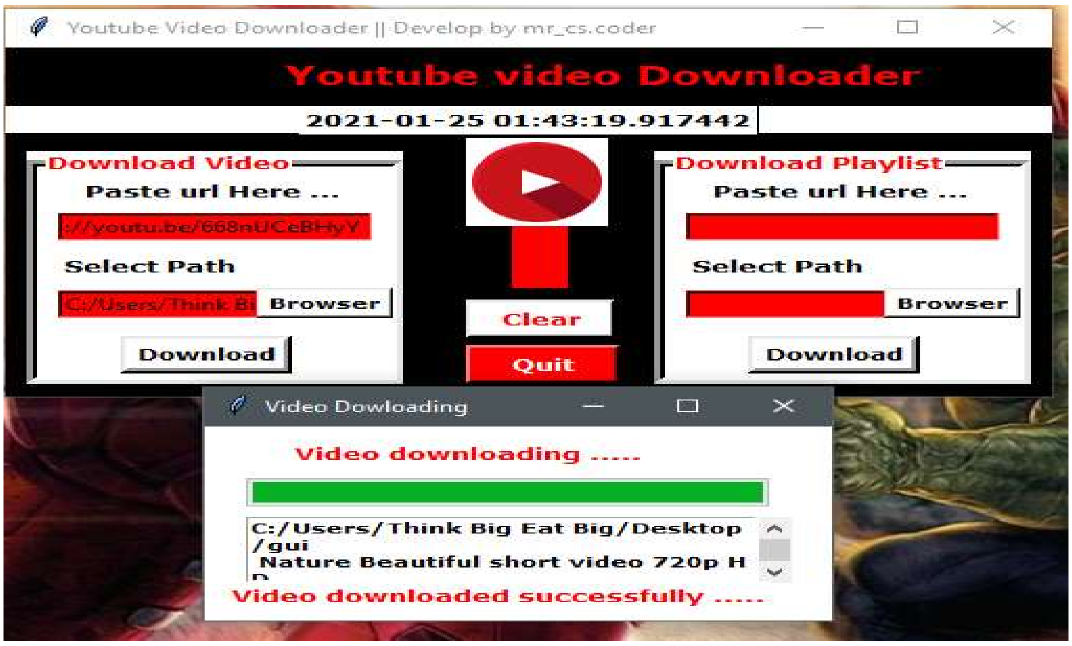
Click Download Button

⇓



Download Compleated

⇩



2021-01-25 01:43:19.917442

://youtu.be/668nUCeBHyY

C:/Users/Think Bi

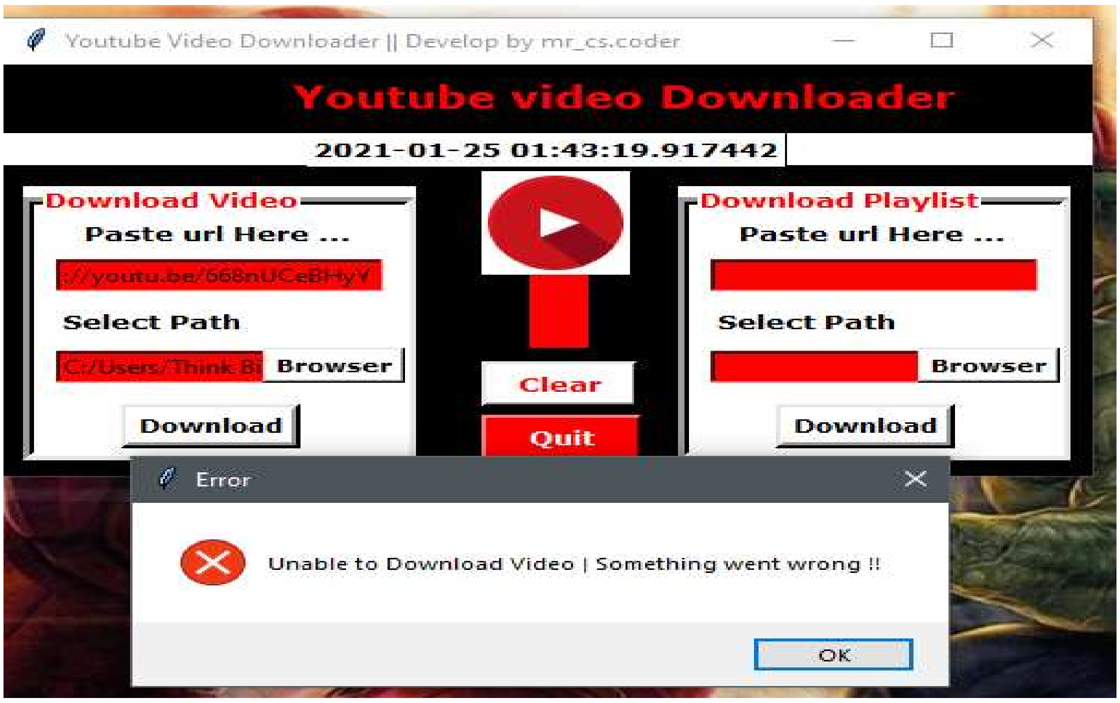
Video downloading .....

C:/Users/Think Big Eat Big/Desktop /gui
Nature Beautiful short video 720p HD

Video downloaded successfully .....

Something Wrong Message

⇩



2021-01-25 01:43:19.917442

://youtu.be/668nUCeBHyY

C:/Users/Think Bi

Unable to Download Video | Something went wrong !!

Advantage

- This application is save your lot of time to download your video and playlist.
- Easy to download all type of video
- Easy to use not so complex GUI interface.

Disadvantage

- Not show download status and download speed .

Scope

- This can be used anywhere for download video and playlist as well as in corporate world.
- Right now not show the download status but in future update this application and add some new feature
- This is used to download video and playlist that make up a path to a particular location.

References

- **website**
  www.python.org
  www.geeksforgeeks.org
- **Youtubes**
- **Book**
  Python Crash Course: A Hands-On, Project-Based Introduction to Programming, Book by Eric Matthes.

CONCLUSION

With this project in python, we have successfully developed the youtube video downloader project using python. We used the popular Tkinter library that used for rendering graphics. We use the pytube library to download videos from youtube. The Youtube downloader project is a python project. The object of this project is to download any type of video in a fast and easy way from youtube in your device.
In this python project, user has to copy the youtube video URL that they want to download and simply paste that URL in the 'paste link here' section and click on the download button, it will start downloading the video. When video downloading finishes, it shows a message 'downloaded' popup on the window below the download button.