REPORT

On

MINI PROJECT (KCS752)

based on

“Color Detection using opencv”

ASHOKA INSTITUTE OF TECNOLOY AND MANAGEMENT (VARANASI)
SESSION (2022-23)

SUBMITTED BY:

NAME: KHUSHI SRIVASTAVA
ROLL NO: 1906410100025
B. Tech (CSE) 4th year

SUBMITTED TO:

Dr. PRITI SINGH
(HOD CS Department)

DECLARATION

I hereby declare that I have completed my six weeks summer training from 01/07/2022 to 13/08/2022 under the guidance of our college. I hereby undertake that the project undertaken by me is the genuine work of mine.

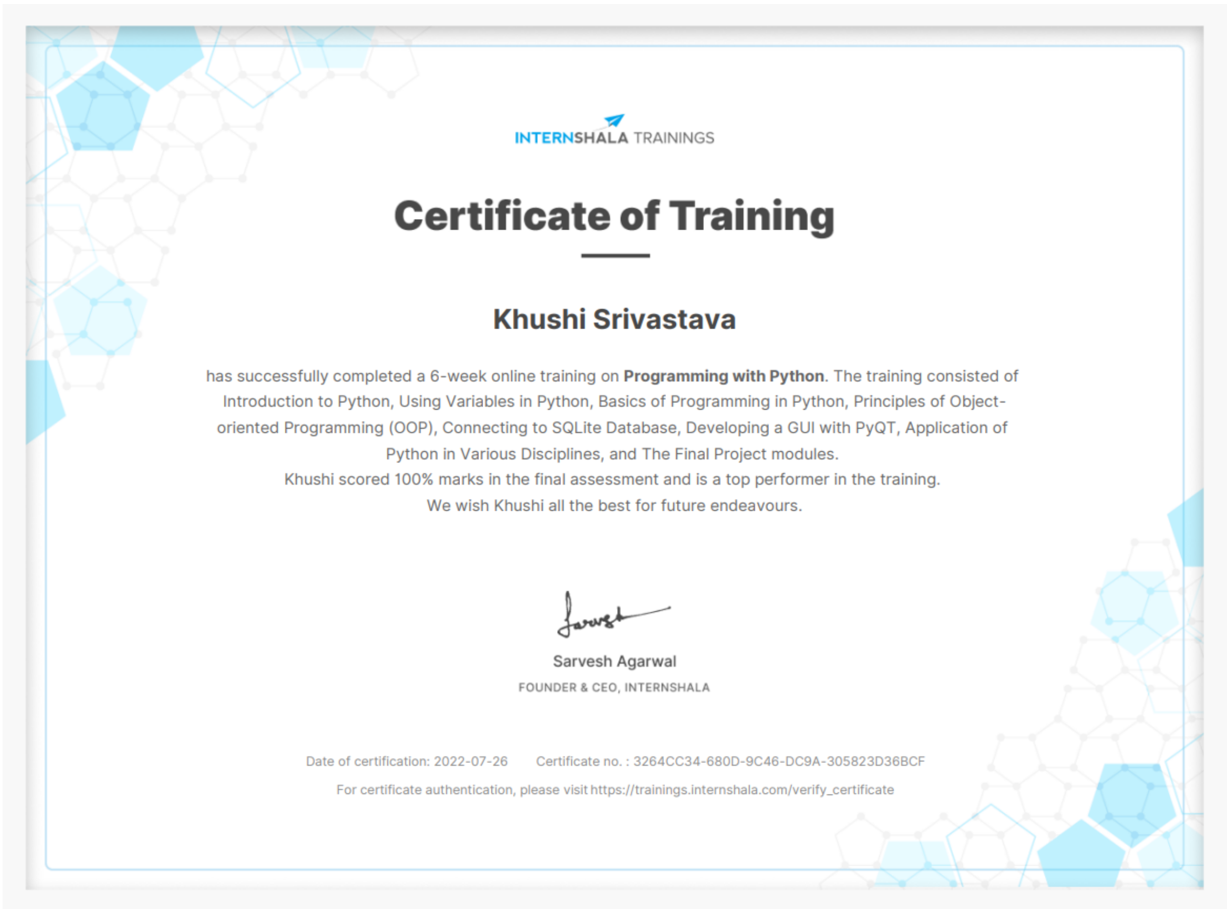
(Signature of student)
Khushi Srivastava
Roll no. 1906410100025

ACKNOWLEDGEMENT

It is my proud privilege and duty to acknowledge the kind of help and guidance received from several people in preparation of this report. It would not have been possible to prepare this report in this form without their valuable help, cooperation and guidance.
First and foremost, I wish to record our sincere gratitude to my Coordinators for their constant support and encouragement in preparation of this report and for making available videos and interface facilities needed to prepare this report.
The seminar on "**Python**" was very helpful to us in giving the necessary background information and inspiration in choosing this topic for the seminar. Their contributions and technical support in preparing this report are greatly acknowledged.

Last but not the least, we wish to thank our parents for financing our studies in this college as well as constantly encouraging us to learn engineering. Their personal sacrifice in providing this opportunity to learn engineering is gratefully acknowledged**.**

SUMMER TRAINING CERTIFICATE



INTERNSHALA TRAININGS

Certificate of Training

Khushi Srivastava

has successfully completed a 6-week online training on **Programming with Python**. The training consisted of Introduction to Python, Using Variables in Python, Basics of Programming in Python, Principles of Object-oriented Programming (OOP), Connecting to SQLite Database, Developing a GUI with PyQT, Application of Python in Various Disciplines, and The Final Project modules.
Khushi scored 100% marks in the final assessment and is a top performer in the training.
We wish Khushi all the best for future endeavours.

[illegible]

Sarvesh Agarwal
FOUNDER & CEO, INTERNSHALA

Date of certification: 2022-07-26 Certificate no. : 3264CC34-680D-9C46-DC9A-305823D36BCF

For certificate authentication, please visit https://trainings.internshala.com/verify_certificate

TRAINING CONTENT

1. Introduction to python
2. Using Variables in python
3. Basics of Programming in python
4. Principle of object-oriented programming
5. Connecting to SQLite Database
6. Developinga GUI with PyQT

Introduction to Python

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. Python interpreters are available for installation on many operating systems, allowing Python code execution on a wide variety of systems.

Scripting Language

A scripting or script language is a programming language that supports scripts, programs written for a special run-time environment that automate the execution of tasks that could alternatively be executed one-by-one by a human operator. Scripting languages are often interpreted (rather than compiled). Primitives are usually the elementary tasks or API calls, and the language allows them to be combined into more complex programs. Environments that can be automated through scripting include software applications, web pages within a web browser, the shells of operating systems (OS), embedded systems, as well as numerous games. A scripting language can be viewed as a domain-specific language for a particular environment; in the case of scripting an application, this is also known as an extension language. Scripting languages are also sometimes referred to as very high-level programming languages, as they operate at a high level of abstraction, or as control languages.

Object Oriented Programming Language

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods. A distinguishing feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). In OO programming, computer programs are designed by making them out of objects that interact with one another. There is significant diversity in object oriented programming, but most popular languages are class-based, meaning that objects are instances of classes, which typically also determines their type.

It simplifies software development and maintenance by providing some concepts:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction

- Encapsulation

Data Type

Data types determine whether an object can do something, or whether it just would not make sense. Other programming languages often determine whether an operation makes sense for an object by making sure the object can never be stored somewhere where the operation will be performed on the object (this type system is called static typing). Python does not do that. Instead it stores the type of an object with the object, and checks when the operation is performed whether that operation makes sense for that object

Python has many native data types. Here are the important ones:

Booleans are either True or False.

Numbers can be integers (1 and 2), floats (1.1 and 1.2), fractions (1/2 and 2/3), or even complex numbers.

Strings are sequences of Unicode characters, e.g. an HTML document.

Bytes and byte arrays, e.g. a JPEG image file. Lists are ordered sequences of values.

Tuples are ordered, immutable sequences of values

Sets are unordered bags of values.

Variable

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in 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 variables, you can store integers, decimals or characters in these variables. Ex: counter = 100 # An integer assignment miles = 1000.0 # A floating point name = "John" # A string.

String

In programming terms, we usually call text a string. When you think of a string as a collection of letters, the term makes sense. All the letters, numbers, and symbols in this book could be a string. For that matter, your name could be a string, and so could your address.

PYTHON OPERATORS

1-Arithmetic operator

Operator	Meaning	Example
+	Add two operands or unary plus	x + y +2

-	Subtract right operand from the left or unary minus	x - y -2
*	Multiply two operands	x * y
/	Divide left operand by the right one (always results into float)	x / y
%	Modulus - remainder of the division of left operand by the right	x % y (remainder of x/y)
//	Floor division - division that results into whole number adjusted to the left in the number line	x // y
**	Exponent - left operand raised to the power of right	x**y (x to the power y)

2-Comparision operator

>	Greater that - True if left operand is greater than the right	x > y
<	Less that - True if left operand is less than the right	x < y
==	Equal to - True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y

>=	Greater than or equal to - True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to - True if left operand is less than or equal to the right	+x <= Y

Tuples

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses.

Accessing Values in Tuples:

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example − tup1 = ('physics', 'chemistry', 1997, 2000); tup2 = (1, 2, 3, 4, 5, 6, 7); print "tup1[0]: ", tup1[0] print "tup2[1:5]: ", tup2[1:5]

When the above code is executed, it produces the following result − tup1[0]: physics

tup2[1:5]: [2, 3, 4, 5]

Basic Tuples Operations

Tuples respond to the + and * operators much like strings; they mean concatenation and repetition here too, except that the result is a new tuple, not a string. In fact, tuples respond to all of the general sequence operations we used on strings in the prior chapter −

Python Expression	**Results**	**Description**
len((1, 2, 3))	3	Length
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenation
('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	Repetition
3 in (1, 2, 3)	True	Membership
for x in (1, 2, 3): print x,	1 2 3	Iteration

List

The list is a most versatile datatype available in Python which can be written as a list of comma- separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example − list1 = ['physics', 'chemistry', 1997, 2000]; list2 = [1, 2, 3, 4, 5]; list3 = ["a", "b", "c", "d"];

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

Accessing Values in Lists:

To access values in lists, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example − list1 = ['physics', 'chemistry', 1997, 2000]; list2 = [1, 2, 3, 4, 5, 6, 7]; print "list1[0]: ", list1[0] print "list2[1:5]: ", list2[1:5]

Output: list1[0]: physics

list2[1:5]: [2, 3, 4, 5]

Update: list = ['physics', 'chemistry', 1997, 2000]; print "Value available at index 2 : " print list[2] list[2] = 2001; print "New value available at index 2 : " print list[2]

Output: Value available at index 2 : 1997 New value available at index 2 : 2001

Delete: list1 = ['physics', 'chemistry', 1997, 2000]; print list1 del list1[2]; print "After deleting value at index 2 : " print list1

['physics', 'chemistry', 1997, 2000]

Output: After deleting value at index 2 : ['physics', 'chemistry', 2000]

Basic List Operation

Python Expression	**Results**	**Description**
len([1, 2, 3])	3	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
3 in [1, 2, 3]	True	Membership
for x in [1, 2, 3]: print x,	1 2 3	Iteration

Python includes following list method

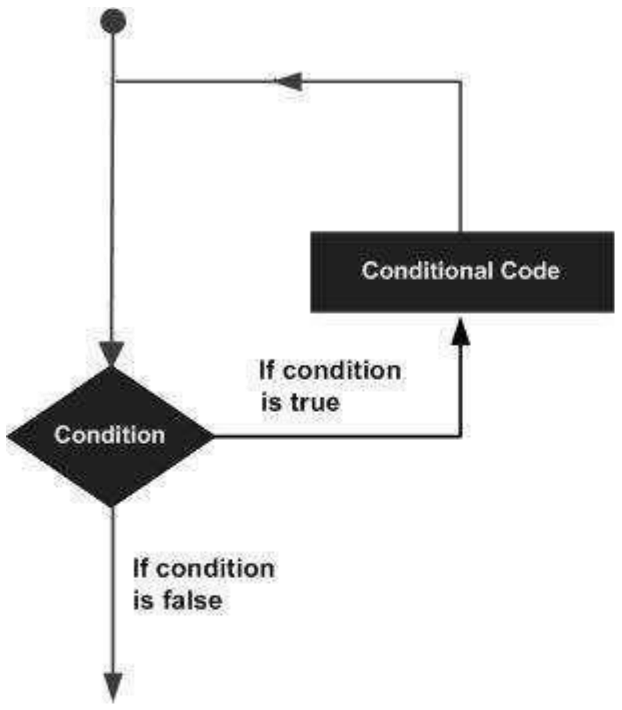
SN	**Methods with Description**
1	**list.append(obj)** Appends object obj to list
2	**list.count(obj)** Returns count of how many times obj occurs in list

3	**list.extend(seq)** Appends the contents of seq to list
4	**list.index(obj)** Returns the lowest index in list that obj appears
5	**list.insert(index, obj)** Inserts object obj into list at offset index
6	**list.pop(obj=list[-1])** Removes and returns last object or obj from list
7	list.remove(obj) Removes object obj from list
8	list.reverse() Reverses objects of list in place
9	list.sort([func]) Sorts objects of list, use compare func if given

Loop definition

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 −



Python programming language provides following types of loops to handle looping requirements.

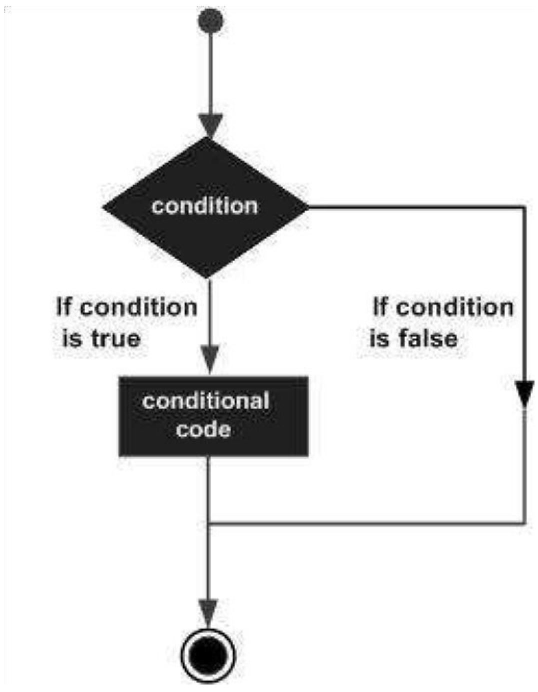
Loop Type	Description
while loop	Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.
for loop	Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.

nested loops	You can use one or more loop inside any another while, for or do..while loop.

Conditional Statements:

Decision making is anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions.

Decision structures evaluate multiple expressions which produce TRUE or FALSE as outcome. You need to determine which action to take and which statements to execute if outcome is TRUE or FALSE otherwise.



Python programming language provides following types of decision making statements. Click the following links to check their detail.

Statement	Description
if statements	An **if statement** consists of a boolean expression followed by one or more statements.

if...else statements	An **if statement** can be followed by an optional **else statement**, which executes when the boolean expression is FALSE.
nested if statements	You can use one **if** or **else if** statement inside another **if** or **else if** statement(s).

Example:

If Statement:

```
a=33

b=20

0

If b>a:

print("b")
```

If...Else Statement*:*

```
a=200

b=33

if b>a:

     print("b is greater than a") else:

   print("a is greater than b")
```

Function

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.

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.

Syntex:

Def

functionname(parameters):

“function_docstring”

Function_suite

Return[expression]

SQLite

Connecting to the Database

Connecting to the SQLite Database can be established using the **connect()** method, passing the name of the database to be accessed as a parameter. If that database does not exist, then it’ll be created.
sqliteConnection = sqlite3.connect('sql.db')

But what if you want to execute some queries after the connection is being made. For that, a cursor has to be created using the **cursor()** method on the connection instance, which will execute our SQL queries.
cursor = sqliteConnection.cursor()

print('DB Init')

The SQL query to be executed can be written in form of a string, and then executed by calling the **execute()** method on the cursor object. Then, the result can be fetched from the server by using the **fetchall()** method, which in this case, is the SQLite Version Number.
query = 'SQL query;'

cursor.execute(query)

result = cursor.fetchall()

print('SQLite Version is {}'.format(result))

Connection class methods

1- Cursor() -A python object that enable us to work with database.
2- Commit()- commits any pending transaction to database
3- Rollback()- cause transaction to be rolledback to starting point.
4- Close()- close connection to the database permanently.

RETREIVING NEW RECORD

1- fetchone() – It give one row value at a time.It is a tuple.
2- fetchall()- fetch all record in the form of list of tuple.

Designing GUI applications Using PyQt in Python

Building GUI applications using the **PYQT designer tool** is comparatively less time-consuming than code the widgets. It is one of the fastest and easiest ways to create GUIs.
The normal approach is to write the code even for the widgets and for the functionalities as well. But using Qt-designer, one can simply drag and drop the widgets, which comes very useful while developing big-scale applications.

SCOPE OF PYTHON

1 - Science

- Bioinformatics

2- System Administration

- Unix
- Web logic
- Web sphere

3- Web Application Development
4- Image processing

CONCLUSION

I believe the trial has shown conclusively that it is both possible and desirable to use Python as the principal teaching language:

- It is Free (as in both cost and source code).
- It is trivial to install on a Windows PC allowing students to take their interest further. For many the hurdle of installing a Pascal or C compiler on a Windows machine is either too expensive or too complicated;
- It is a flexible tool that allows both the teaching of traditional procedural programming and modern OOP; It can be used to teach a large number of transferable skills;
- It is a real-world programming language that can be and is used in academia and the commercial world;
- It appears to be quicker to learn and, in combination with its many libraries, this offers the possibility of more rapid student development allowing the course to be made more challenging and varied;

and most importantly, its clean syntax offers increased understanding and enjoyment for students.

The training program having three destinations was a lot more useful than staying at one place throughout the whole 6 weeks. In my opinion, I have gained lots of knowledge and experience needed to be successful in great engineering challenge as in my opinion; Engineering is after all a challenge and not a job.