**A SUMMER TRAINING**

**REPORT**

**On**

**ADVANCED PYTHON PROGRAMMING LANGUAGE by TECHVANTO ACADEMY**

Submitted in partial fulfilment of the requirements for the award of degree of B. Tech (Computer Science & Engineering)

**SUBMITTED TO :**

**LOVELY PROFESSIONAL UNIVERSITY**

**PHAGWARA , PUNJAB**

****

**From 25th May’ 2022 – 10th July’2022**

**SUBMITTED BY :**

**SOUVIK CHAKRABORTY**

**Registration No. : 12011247**

**Under the guidance of**

**SHIVA SIR**

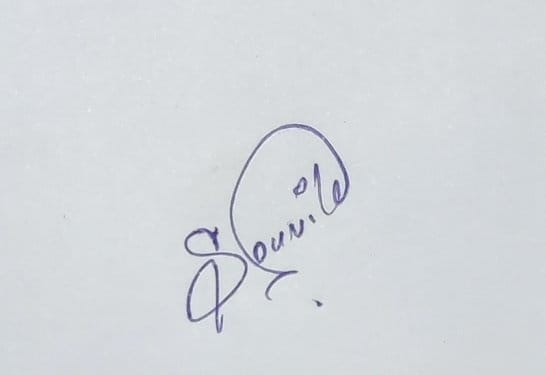
**Student Declaration**

**To whom so ever it may concern**

I Souvik Chakraborty , 12011247 ,hereby declare that the work done by me on“ADVANCED PYTHON PROGRAMMING LANGUAGE” from 25th May’ 2022 – 10th July’2022 is a record of original work for the partial fulfillment of the requirement for the award of the degree , Bachelors of Technology (CSE)

Name of the student : Souvik Chakraborty

Regd . No. : 12011247

Signature of the student : 

Dated : July 19, 2022

* **Summer Training Certificate from Techvanto Academy:**

****

**ACKNOWLEDGEMENT**

It is our honour and duty to express our gratitude for the assistance and direction that many people provided in the preparation of this report. Without their invaluable assistance, cooperation, and direction, it would not have been possible to prepare this report in the manner that it was.

Prior to anything else, we want to express our sincere gratitude to family and friends for their unwavering support and inspiration during the writing of this report.

The "Advanced Python Programming " seminar was very beneficial to us in that it provided the background knowledge and inspiration we needed to select this topic for the seminar. We gratefully acknowledge their contributions and technical assistance in the creation of this report.

Last but not least, we would like to thank our parents for supporting our college education and for always pushing us to pursue engineering. We sincerely thank them for making the sacrifice of giving us this chance to learn engineering.

**Thank You!**

Table Of Contents

1.Introduction :

|  |  |  |
| --- | --- | --- |
| 1.1 | Python | 1-2 |
| 1.2 | Scripting Language | 2-3 |
| 1.3 | Object Oriented Programming | 3-3 |
| 1.4 | History of python | 3-4 |
| 1.5 | Behind the Scene of Python | 5-5 |

2.Downloading & Installing Python :

|  |  |  |
| --- | --- | --- |
| 2.1 | Downloading Python | 6-7 |
| 2.2 | Installing Python | 8-10 |
| 2.3 | Setup Path of Variable | 10-11 |
| 2.4 | Running The Python IDE | 12-13 |
| 2;.5 | Python Code Execution | 13-13 |

3.Data Types & Operator :

|  |  |  |
| --- | --- | --- |
| 3.1 | Data Type | 14-15 |
| 3.2 | Variables | 15-15 |
| 3.3 | String | 15-16 |
| 3.4 | Python Operator | 16-17 |
| 3.4.1 | Arithmetic Operator | 17-17 |
| 3.4.2 | Comparison Operator | 17-18 |

4.Tuple & List :

|  |  |  |
| --- | --- | --- |
| 4.1 | Tuple | 18-20 |
| 4.1.1 | Accessing Tuple Values | 18 -18 |
| 4.1.2 | Built in Operation | 18-19 |
| 4.1.3 | Built in Tuple Functions | 19-19 |
| 4.2 | List | 20-23 |
| 4.2.1 | Accessing List Values | 20-21 |
| 4.2.2 | Built in Operation | 21-21 |
| 4.2.3 | Built in Functions | 21-21 |

5.Loops & Conditional Statements :

|  |  |  |
| --- | --- | --- |
| 5.1 | Loops | 21-26 |
| 5.1.1 | Loops Definition | 21-22 |
| 5.1.2 | Loops Example | 22-23 |
| 5.2 | Conditional Statement | 23-23 |
| 5.2.1 | Conditional Statement Definition | 23-23 |
| 5.2.2 | Conditional Statement Example | 24-25 |
| 5.3 | Function | 25-25 |
| 5.3.1 | Syntax & Examples | 25-26 |

6. Uses & Scope of Python :

|  |  |  |
| --- | --- | --- |
| 6.1 | What Can we do with python ? | 26-26 |
| 6.2 | Who uses python today? | 27-27 |
| 6.3 | Why do people use python ? | 27-27 |

7. MINI PROJECTS

|  |  |  |
| --- | --- | --- |
| 7.1 | ATM & BANK PROTOTYPE | 28-28 |
| 7.2 | CAB BOOKING SYSTEM | 28-28 |
| 7.3 | E.V.M OR V.V.P.A.T MACHINE | 29-29 |
| 7.4 | SCIENTIFIC CALCULATOR | 29-29 |

# Python

A popular high-level, all-purpose, interpreted, dynamic programming language is Python. Programmers can express concepts in less code thanks to its syntax and design philosophy, which places an emphasis on code readability, than they would be able to in languages like C++ or Java. The language offers constructs designed to support both small- and large-scale, concise programmes.

Python supports a variety of programming paradigms, including imperative, functional, and object-oriented or procedural programming. It has a large and thorough standard library, a dynamic type system, automatic memory management, and more. Python code can be executed on a variety of operating systems thanks to the availability of Python interpreters for installation.

# Scripting Language

A programming language that supports scripts is known as a scripting language. Scripts are programmes created for a unique run-time environment that automate the completion of tasks that would otherwise need to be completed one at a time by a human operator.

Sometimes, scripting languages are interpreted (rather than compiled). Primitives are typically the simpler operations or API requests, and the language enables their combination into more intricate programmes. Scripting can automate a variety of environments, including software programmes, web pages viewed in a web browser, operating system (OS) shells, embedded systems, and many games.

A scripting language, also known as an extension language when used to script an application, can be thought of as a domain-specific language for a specific environment. Due to their high level of abstraction, scripting languages are also sometimes referred to as control languages or very high-level programming languages.

# Object Oriented Programming Language

A programming paradigm known as object-oriented programming (OOP) is based on the idea of "objects," which can include both code and data in the form of procedures, which are frequently referred to as methods and fields. The ability of an object's procedures to access and frequently modify the data fields of the object with which it is associated is a defining characteristic of objects (objects have a notion of "this" or "self").

Computer programmes are created using object-oriented programming (OO) by constructing them from objects that communicate with one another. Although there is a lot of variety in object-oriented programming, the majority of widely used programming languages are class-based, which means that objects are instances of classes, and classes typically also define an object's type.

# History

The idea for Python originated in the late 1980s, and Guido van Rossum began working on it at CWI in the Netherlands in December 1989 as a replacement for the ABC language—which was in turn inspired by SETL—capable of handling exceptions and interacting with the Amoeba operating system. Van Rossum is the primary author of Python, and the Python community has given him the title of "benevolent dictator for life" to reflect his continued influence over the language's direction (BDFL).

“Python is an experiment in how much freedom programmers need. Too much freedom and nobody can read another's code; too little and expressiveness is endangered.”

- Guido van Rossum



# Downloading python

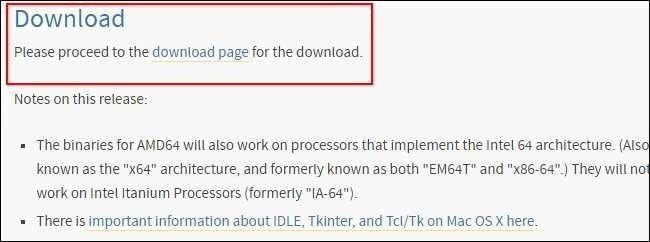
If you don’t already have a copy of Python installed on your computer, you will need to open up your Internet browser and go to the Python download page ***(***[***http://www.python.org/download/).***](http://www.python.org/download/))



Now that you are on the download page, select which of the software builds you would like to download. For the purposes of this article we will use the most up to date version available (Python 3.4.1).



Once you have clicked on that, you will be taken to a page with a description of all the new updates and features of 3.4.1, however, you can always read that while the download is in process. Scroll to the bottom of the page till you find the “Download” section and click on the link that says “download page.”



Now you will scroll all the way to the bottom of the page and find the “Windows x86 MSI installer.” If you want to download the 86-64 bit MSI, feel free to do so. We believe that even if you have a 64-bit operating system installed on your computer, the 86-bit MSI is preferable. We say this because it will still run well and sometimes, with the 64- bit architectures, some of the compiled binaries and Python libraries don’t work well.



# Installing Python

Once you have downloaded the Python MSI, simply navigate to the download location on your computer, double clicking the file and pressing Run when the dialog box pops up.

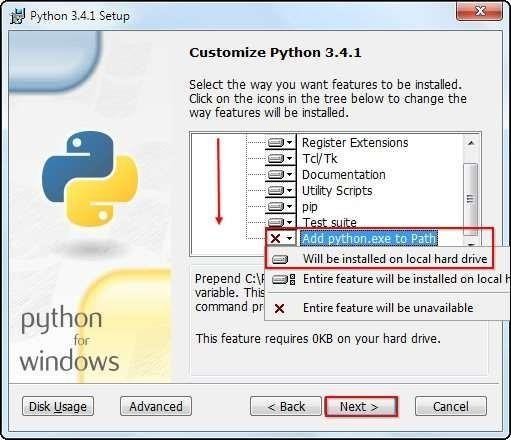


If you are the only person who uses your computer, simply leave the “Install for all users” option selected. If you have multiple accounts on your PC and don’t want to install it across all accounts, select the “Install just for me” option then press “Next.”



f you want to change the install location, feel free to do so; however, it is best to leave it as is and simply select next, Otherwise...

Scroll down in the window and find the “Add Python.exe to Path” and click on the small red “x.” Choose the “Will be installed on local hard drive” option then press “Next.”



Now that you have completed the installation process, click on “Finish.

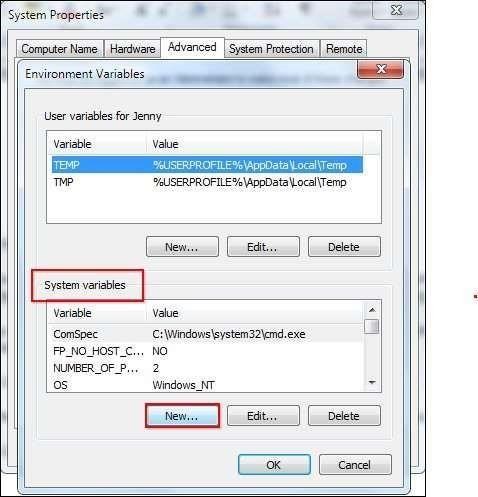


# Setup the Path Variable

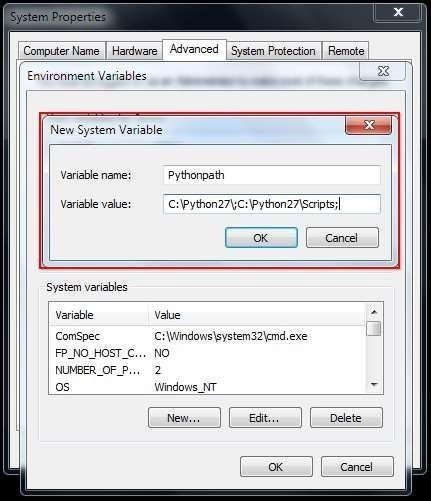
Begin by opening the start menu and typing in “environment” and select the option called “Edit the system environment variables.”

When the “System Properties” window appears, click on “Environment Variables…”

Once you have the “Environment Variables” window open, direct your focus to the bottom half. You will notice that it controls all the “System Variables” rather than just this associated with your user. Click on “New…” to create a new variable for Python.



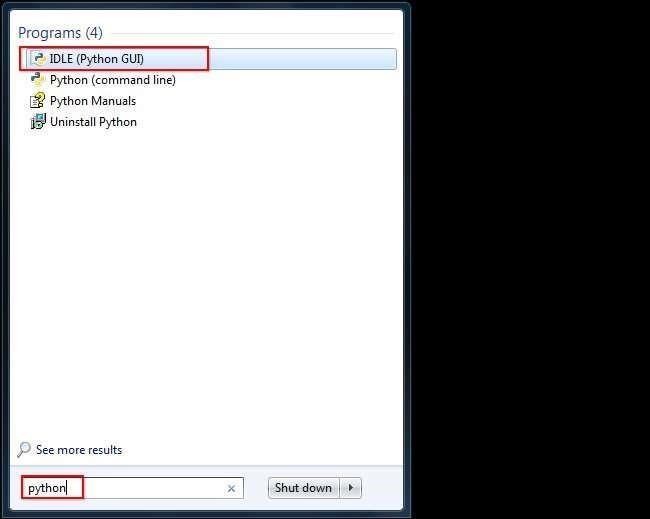
Simply enter a name for your Path and the code shown below. For the purposes of this example we have installed Python 2.7.3, so we will call the path: “Pythonpath.” The string that you will need to enter is: “C:\Python27\;C:\Python27\Scripts;”



# Running The Python IDE

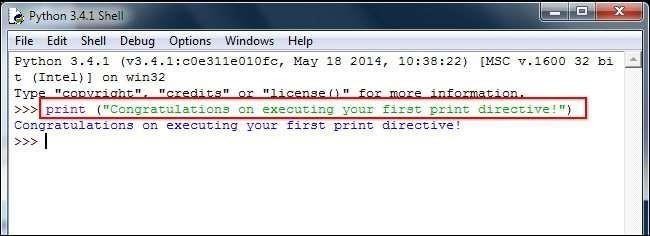
Now that we have successfully completed the installation process and added our “Environment Variable,” you are ready to create your first basic Python script. Let’s begin by opening Python’s GUI by pressing “Start” and typing “Python” and selecting the “IDLE

(Python GUI).”



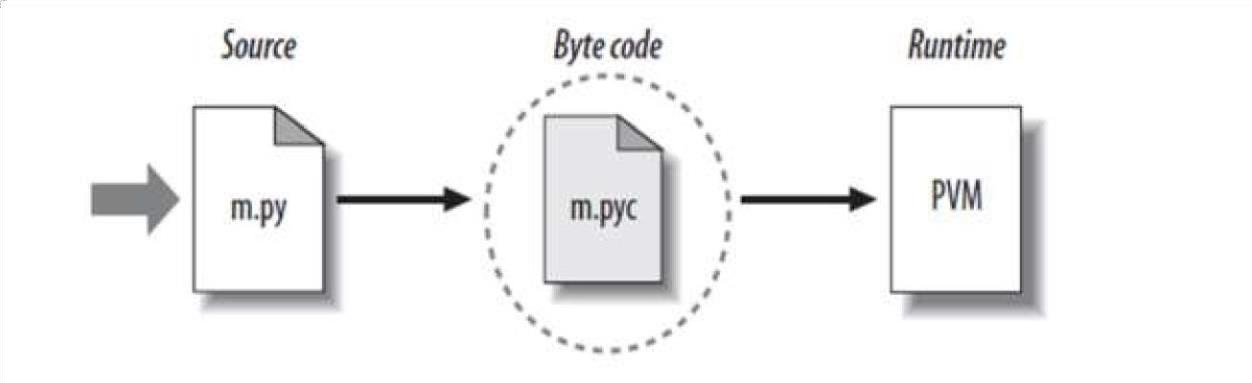
Once the GUI is open, we will begin by using the simplest directive possible. This is the “print” directive which simply prints whatever you tell it to, into a new line. Start by typing a print directive like the one shown in the image below or copy and paste this text then press

“Enter”: print (“Congratulations on executing your first print directive!”)



Python Code Execution

Python’s traditional runtime execution model: source code you type is translated to byte code, which is then run by the Python Virtual Machine. Your code is automatically compiled, but then it is interpreted.



Source code extension is .py

Byte code extension is .pyc (compiled python code)

# Data Type

It's known as dynamic typing. Whether an object can do something or if it simply makes no sense depends on its data type. By ensuring that an object can never be stored in a location where the operation will be carried out on it, other programming languages frequently determine whether an operation makes sense for an object (this type system is called static typing). Python avoids doing that. Instead, it stores the type of an object with the object and determines whether the operation is appropriate for that object before performing it.

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 merely reserved memory spaces for the storage of values. This implies that you set aside some memory when you create a variable.

The interpreter allots memory and determines what can be stored in the reserved memory based on the data type of a variable. Therefore, you can store integers, decimals, or characters in these variables by giving them different data types.

Ex: counter = 100 # An integer

assignment miles = 1000.0 # A floating

point name= "John" # A string

# String

Text is commonly referred to as a "string" in programming. The term makes sense when you consider a string to be a collection of letters.

This book's alphabet, numeric code, and symbols are all possible strings. In fact, both your name and your address could be strings.

# Creating Strings

In Python, we create a string by putting quotes around text. For example, we could take our otherwise useless

|  |  |  |
| --- | --- | --- |
| • | "hello"+"world" | "helloworld" # concatenation |

|  |  |  |  |
| --- | --- | --- | --- |
| • | "hello"\*3 | "hellohellohello" | # repetition |
| • | "hello"[0] | "h" | # indexing |
| • | "hello"[-1] | "o" | # (from end) |
| • | "hello"[1:4] | "ell" | # slicing |
| • | len("hello") | 5 | # size |
| • | "hello" < "jello" | 1 | # comparison |
| • | "e" in "hello" | 1 | # search |

# Python Operator 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) |

**Comparison Operator**

|  |  |  |  |
| --- | --- | --- | --- |
| > | Greater that - True if left operand is greater than the right | x > y | **Cha** |
|  |  |  |
|  | | |
| < | 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

Sequence of immutable Python objects make up a tuple. Like lists, tuples are sequences. Tuples and lists differ in that tuples can't 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 TuplesOperations

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 |

Built-in Tuple Functions

Python includes the following tuple functions −

|  |  |
| --- | --- |
| **SN** | **Function with Description** |
| 1 | [**cmp(tuple1, tuple2)**](http://www.tutorialspoint.com/python/tuple_cmp.htm) Compares elements of both tuples. |
| 2 | [**len(tuple)**](http://www.tutorialspoint.com/python/tuple_len.htm) Gives the total length of the tuple. |
| 3 | [**max(tuple)**](http://www.tutorialspoint.com/python/tuple_max.htm) Returns item from the tuple with max value. |
| 4 | [**min(tuple)**](http://www.tutorialspoint.com/python/tuple_min.htm) Returns item from the tuple with min value. |
| 5 | [**tuple(seq)**](http://www.tutorialspoint.com/python/tuple_tuple.htm) Converts a list into tuple. |

# List

The list datatype, one of the most flexible ones offered by Python, can be expressed as a list of values (items) separated by commas and enclosed in square brackets. The fact that a list's items do not have to be of the same type is crucial..

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 ListOperation

|  |  |  |
| --- | --- | --- |
| **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 |

Built-in List Functions & Methods:

|  |  |
| --- | --- |
| **SN** | **Function with Description** |
| 1 | [**cmp(list1, list2)**](http://www.tutorialspoint.com/python/list_cmp.htm) Compares elements of both lists. |
| 2 | [**len(list)**](http://www.tutorialspoint.com/python/list_len.htm) Gives the total length of the list. |
| 3 | [**max(list)**](http://www.tutorialspoint.com/python/list_max.htm) Returns item from the list with max value. |
| 4 | [**min(list)**](http://www.tutorialspoint.com/python/list_min.htm) Returns item from the list with min value. |
| 5 | [**list(seq)**](http://www.tutorialspoint.com/python/list_list.htm) Converts a tuple into list. |

Python includes following list methods

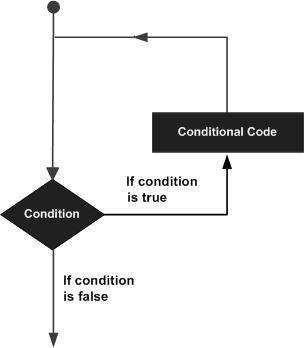
|  |  |
| --- | --- |
| **SN** | **Methods with Description** |
| 1 | [**list.append(obj)**](http://www.tutorialspoint.com/python/list_append.htm) Appends object obj to list |
| 2 | [**list.count(obj)**](http://www.tutorialspoint.com/python/list_count.htm) Returns count of how many times obj occurs in list |
| 3 | [**list.extend(seq)**](http://www.tutorialspoint.com/python/list_extend.htm) Appends the contents of seq to list |
| 4 | [**list.index(obj)**](http://www.tutorialspoint.com/python/list_index.htm) Returns the lowest index in list that obj appears |
| 5 | [**list.insert(index, obj)**](http://www.tutorialspoint.com/python/list_insert.htm) Inserts object obj into list at offset index |
| 6 | [**list.pop(obj=list[-1])**](http://www.tutorialspoint.com/python/list_pop.htm) Removes and returns last object or obj from list |

|  |  |
| --- | --- |
| 7 | [**list.remove(obj)**](http://www.tutorialspoint.com/python/list_remove.htm) Removes object obj from list |
| 8 | [**list.reverse()**](http://www.tutorialspoint.com/python/list_reverse.htm) Reverses objects of list in place |
| 9 | [**list.sort([func])**](http://www.tutorialspoint.com/python/list_sort.htm) Sorts objects of list, use compare func if given |

# Loop definition

Different control structures offered by programming languages enable more intricate execution paths.

We can run a statement or set of statements repeatedly by using a loop statement.The following diagram illustrates a loop statement −



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

|  |  |
| --- | --- |
| **Loop Type** | **Description** |

|  |  |
| --- | --- |
| [**while loop**](http://www.tutorialspoint.com/python/python_while_loop.htm) | Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body. |
| [**for loop**](http://www.tutorialspoint.com/python/python_for_loop.htm) | Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable. |
| [**nested loops**](http://www.tutorialspoint.com/python/python_nested_loops.htm) | You can use one or more loop inside any another while, for or do..while loop. |

Loop Example:

For Loop:

>>> for mynum in [1, 2, 3, 4,

5]: print ("Hello", mynum ) Hello 1

Hello 2

Hello 3

Hello 4

Hello 5 While Loop:

>>> count = 0 >>while(count< 4):

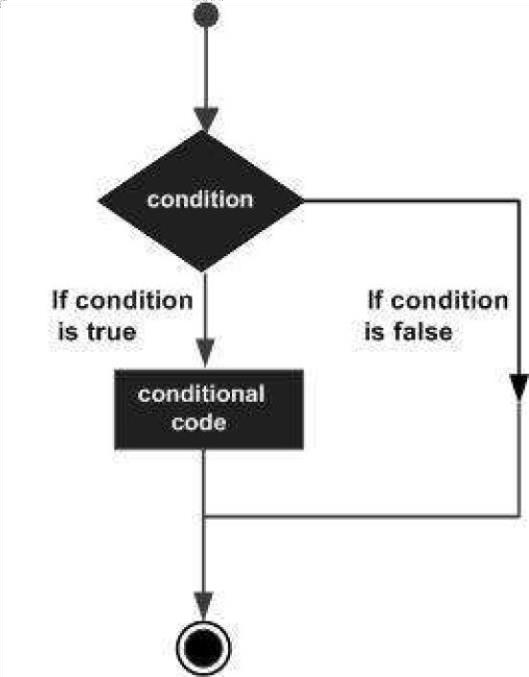
print 'The count is:', count count = count + 1

The count is: 0 The count is:

1 The count is: 2 The count is: 3

Conditional Statements:

A decision is the specification of actions to be taken in response to conditions that may arise during programme execution.

Multiple expressions are evaluated by decision structures, and the results are TRUE or FALSE. If the outcome is TRUE or FALSE, you must decide which statement to execute and which action to take.

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

|  |  |
| --- | --- |
| **Statement** | **Description** |
| [**if statements**](http://www.tutorialspoint.com/python/python_if_statement.htm) | An **if statement** consists of a boolean expression followed by one or more statements. |
| [**if...else statements**](http://www.tutorialspoint.com/python/python_if_else.htm) | An **if statement** can be followed by an optional **else statement**, which executes when the boolean expression is FALSE. |
| [**nested if statements**](http://www.tutorialspoint.com/python/nested_if_statements_in_python.htm) | 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

# The word "def" precedes the function name and parentheses (()) in a function block.

# These parentheses are where you should put any arguments or input parameters. Within these parentheses, parameters can also be defined.

# The function's documentation string may be an optional first statement in a function.

# Every function has an indented code block that begins with a colon (:).

# Returning an expression to the caller is possible when leaving a function using the return statement. Return None is the same as a return statement with no arguments.Syntex:

Def functionname(parameters): “function\_docstring” Function\_suite Return[expression]

# Example:

Def printme(str):

“this print a passed string into this function” print str return

1. *# Function definition is here*

def printme( str ):

"This prints a passed string into this function" print str return;

*# Now you can call printme function* printme("I'm first call to user defined function!") printme("Again second call to the same function")

SCOPE OF PYTHON

1. - Science
   * Bioinformatics
2. - System Administration
   * Unix
   * Web logic
   * Web sphere
3. - Web Application Development

# What Can We do With Python?

1. - System programming
2. - Graphical User Interface Programming 3 - Internet Scripting 4 - Component

Integration 5 - Database Programming

6 - Gaming, Images, XML , Robot and more

WHO USES PYTHONTODAY?

* Python is being applied in real revenue-generating products by real companies.
* Google makes extensive use of Python in its web search system, and employs Python’s creator.
* Intel, Cisco, Hewlett-Packard, Seagate, Qualcomm, and IBM use Python for hardware testing.
* ESRI uses Python as an end-user customization tool for its popular GIS mapping products.

WHY DO PEOPLE USE PYTHON?

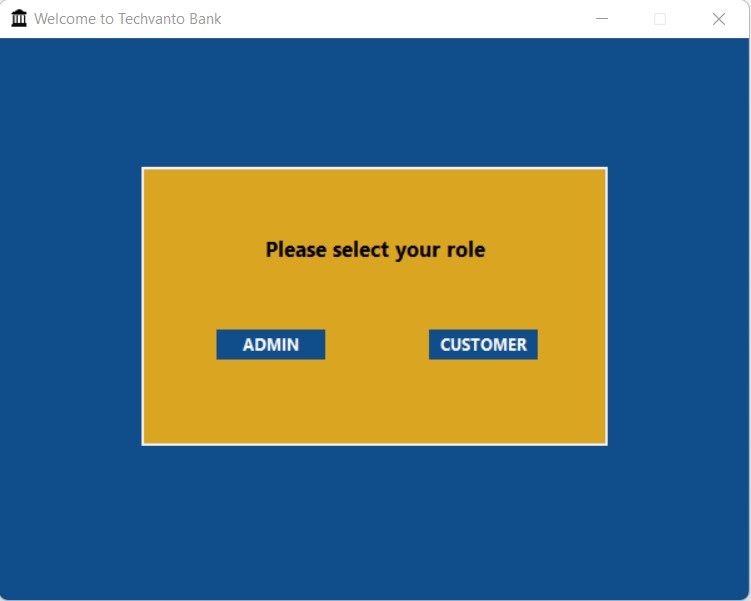
* The YouTube video sharing service is largely written in Python.
* The object-oriented nature of Python's structure makes it compatible with ideas like polymorphism, operation overloading, and multiple inheritance.
* Indentation oIndentation is one of the greatest future in Python.
* Python is easy to download and install, and it is free (open source). oSource code is readily available.
* It's powerful o Dynamic typing o Built-in types and tools o Library utilities

o Third party utilities (e.g. Numeric, NumPy, SciPy) o Automatic memory management

* It's transportable Python runs on almost all of the popular platforms in use today. Python programmes will run identically on any platform as long as you have a compatible Python interpreter installed.

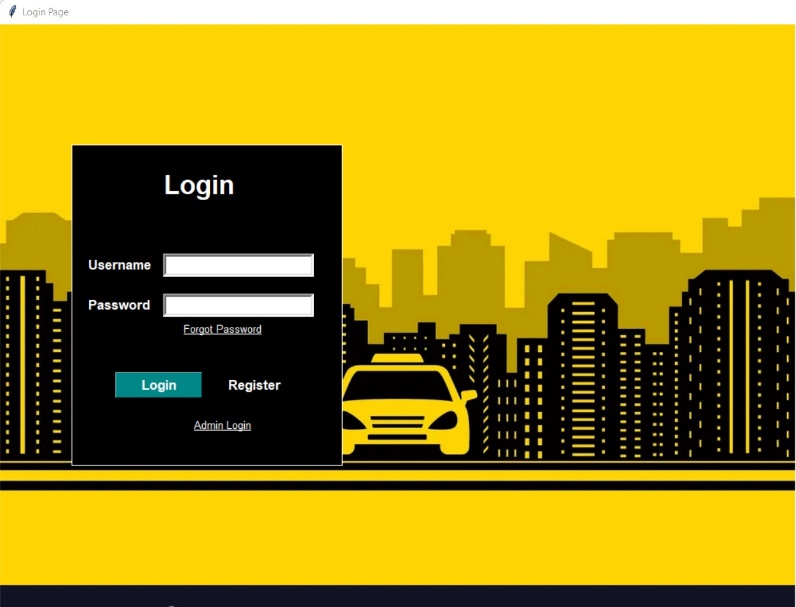
**MINI PROJECTS**

* ATM & BANK PROTOTYPE:



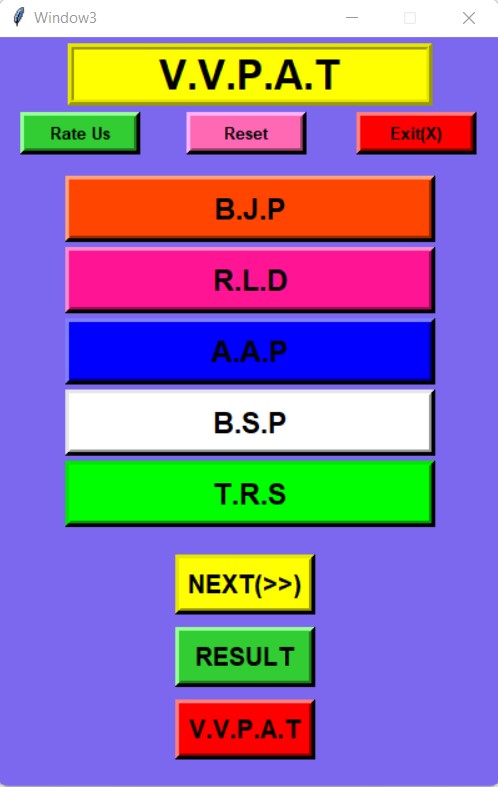
1STPAGE

* CAB BOOKING SYSTEM



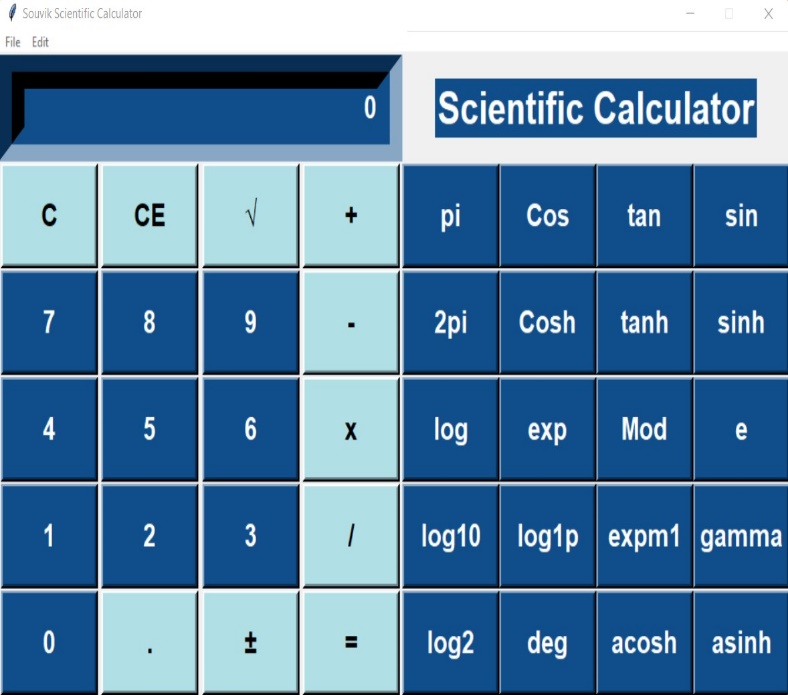
1STPAGE

* E.V.M OR V.V.P.A.T MACHINE



2NDPAGE

* SCIENTIFIC CALCULATOR



2NDPAGE

# Conclusion

* In my opinion, the trial has conclusively demonstrated that using Python as the primary teaching language is both feasible and desirable:
* It's Free (as in both cost and source code).
* The ease of installation on a Windows PC enables students to pursue their interests further. The challenge of installing a Pascal or C compiler on a Windows machine is too expensive or difficult for many people.
* It is a versatile tool that enables the teaching of both conventional procedural programming and contemporary object-oriented programming, and it can be used to impart a wide range of transferrable skills.
* It is a practical programming language that can be used in both the academic and business sectors.
* It seems to be easier to pick up, and when combined with its extensive library, this presents the opportunity for more rapid student growth, enabling the course to be made more challenging and interesting and most importantly, its clean syntax offers increased understanding and enjoyment for students.