PYTHON

A

Training Report

submitted

in partial fulfillment

for the award of the Degree of

Bachelor of Technology

in

Department of Computer Science Engineering

(With Specialization In Computer Science & Engineering)

Supervisor:

Mrs. Amrata Pupneja

Assistant Professor

Submitted By:

Piyush Hariyani

Roll No.22EMICS047

Department of Computer Science & Engineering

Modi Institute of Technology

Rajasthan Technical University

September, 2024 -2025

CANDIDATE'S DECLARATION

I hereby declare that the work, which is being presented in the Training Report, entitled **"Python"** in partial fulfillment for the award of Degree of "Bachelor of Technology" in **Department of Computer Engineering with specialization in Computer Science and Engineering**, and submitted to the **Department of Computer Science & Engineering, Modi Institute of Technology**, Rajasthan Technical University is a record of my own investigations carried under the Guidance of Assistant Professor, Mrs. Amrata Pupneja , Dept. of **Computer Science & Engineering**.

I have not submitted the matter presented in this Training Report anywhere for the award of any other Degree.

Piyush Hariyani

Computer Science and Engineering

Roll No.: 22EMICS047

Modi Institute of Technology, Kota

Counter Signed By:

Supervisor:

Mrs. Amrata Pupneja

Assistant Professor, CSE Dept.

Acknowledgment

I would also like to give my special thanks to the Principal, **Dr. Vikas Soni**, Modi Institute of Technology, for providing the opportunity to me to undertake this work.

I would like to thank my guide Assistant Professor, Mrs. Amrata Pupneja, for their valuable guidance. I appreciate their presence for all the discussions, suggestions and their time whenever I needed them.

Two Persons who deserves a First and Foremost mention are my Mother **Mrs. Damini Hariyani** and My Father **Mr. Hemant Hariyani**, whose strong belief in my abilities and moral support uplifted my spirits. Without their encouragement, I would have never imagined to achieve this height in my career.

I cannot forget to mention the name of my best friends for her relentless help and motivation all through.

Finally, I would like to thank everybody who was important to the successful realization of this report, as well as expressing my apology that I could not mention them personally one by one.

Piyush Hariyani

Computer Science and Engineering

Roll No.: 22EMICS047

Modi Institute of Technology

CERTIFICATE

This is to certify that this Training Report entitled “**Python**” has been successfully carried out by **Piyush Hariyani (Enrolment No.:** 22E1MICSM40P047**),** under my supervision and guidance, in partial fulfillment of the requirement for the award of **Bachelor of Technology** Degree in Computer Science & Engineering from **Modi Institute of Technology, Kota.**

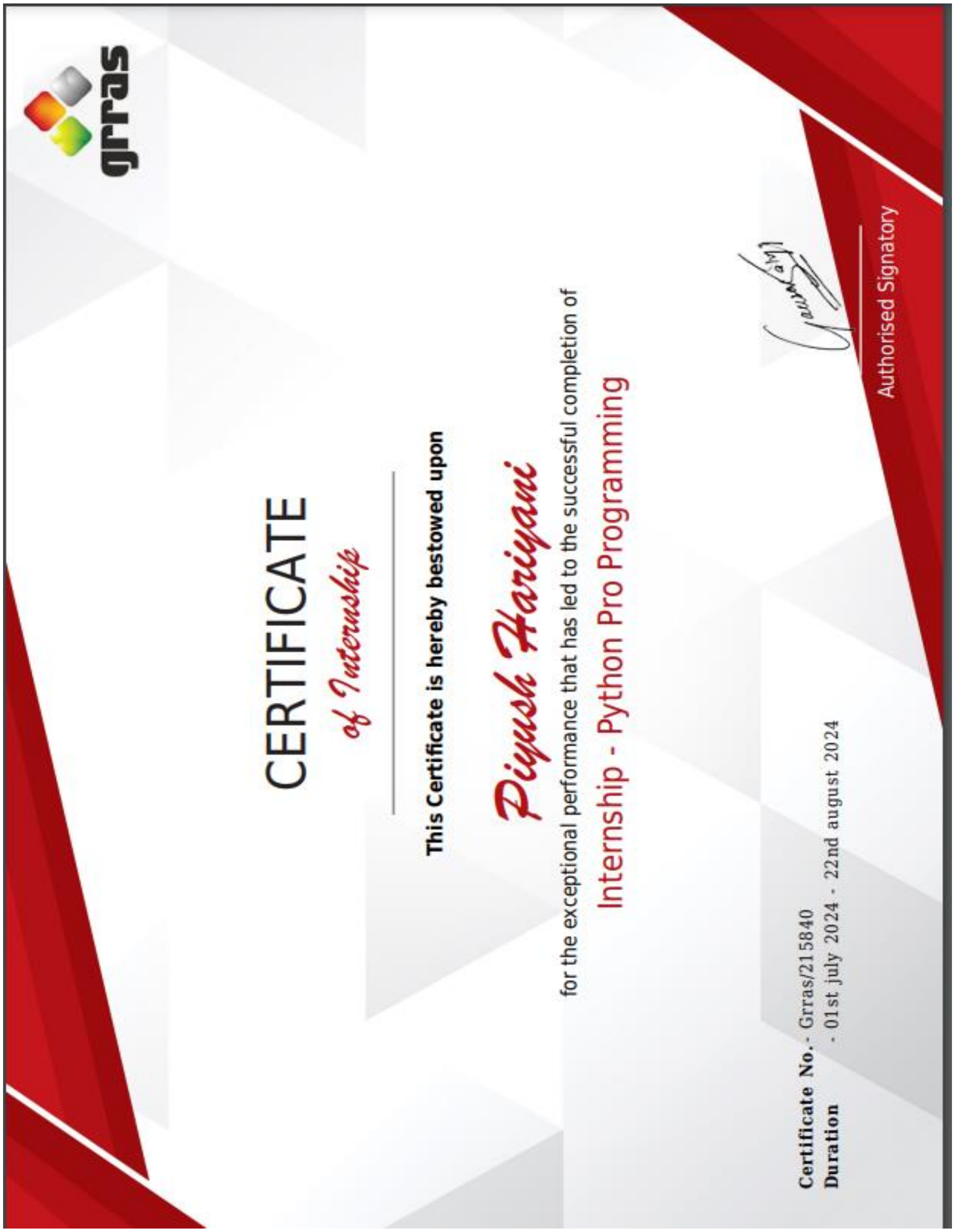
Supervisor:

Mrs. Amrata Pupneja

Assistant Professor, CSE Dept

Place: Kota

Date:



grras

CERTIFICATE

of Internship

This Certificate is hereby bestowed upon

Piyush Hariyani

for the exceptional performance that has led to the successful completion of

Internship - Python Pro Programming

[illegible]

Authorised Signatory

Certificate No. - Grras/215840

Duration - 01st july 2024 - 22nd august 2024

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The objective of a practical training is to learn something about industries practically and to be familiar with a working style of a technical worker to adjust simply according to industrial environment .

This report deals with the equipments their relation and their general operating principle. Python, an interpreted language which was developed by Guido van Rossum came into implementation in 1989. The language supports both object oriented and procedure oriented approach.

Python is designed to be a highly extensible language. Python works on the principle of “there is only one obvious way to do a task” rather than “there is more than one way to solve a particular problem”.

Python is very easy to learn and implement. The simpler syntax, uncomplicated semantics and approach with which Python has been developed make very easier to learn. A large number of python implementations and extensions have been developed since its inception.

Learning Python can open up numerous career opportunities. It’s one of the most sought-after languages by employers due to its applicability in various domains like web development, data analysis, artificial intelligence, automation, and more.

Companies like Google, Netflix, and NASA rely on Python for critical parts of their operations, making it a valuable skill in the job market.

CHAPTER 1

INTRODUCTION

1.1 Python:-

Python is a widely used high-level, general-purpose, interpreted, dynamic programming The language provides constructs intended to enable clear programs on both a small and large scale. 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. 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.

1.2 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.

Python is a computer programming language often used to build websites and software, automate tasks and conduct data analysis. Python is a general purpose language, meaning it can be used to create a variety of different program and isn't specialized for any specific problem to do so.

1.3 Object Oriented Programming:-

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 classbased, meaning that objects are instances of classes, which typically also determines their type.

1.4 History:-

Python was conceived in the late 1980s, and its implementation was started in December 1989 by Guido van Rossum at CWI in the Netherlands as a successor to the ABC language (itself inspired by SETL) capable of exception handling and interfacing with the Amoeba operating system. Van Rossum is Python's principal author, and his continuing central role in deciding the direction of Python is reflected in the title given to him by the Python community, benevolent dictator for life (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

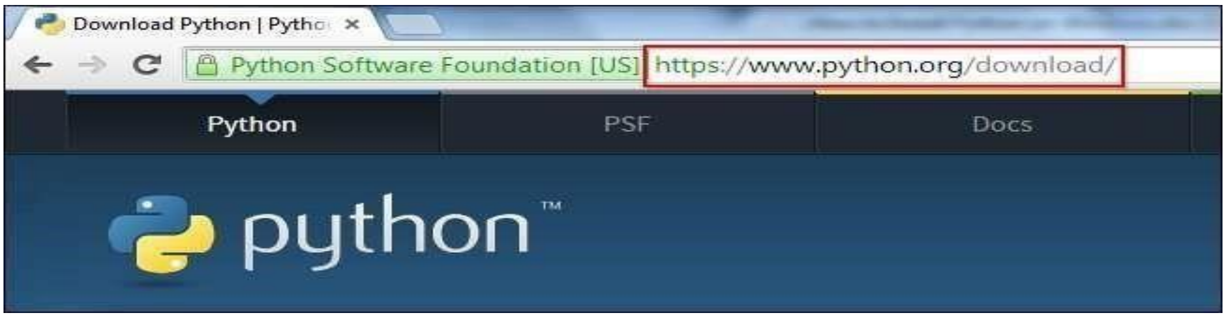
CHAPTER 2

DOWNLOADING PYTHON

2.1 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/).



Download Python | Pytho
Python Software Foundation [US]
https://www.python.org/download/
Python
PSF
Docs
python™

Fig:2.1: downloading page of python

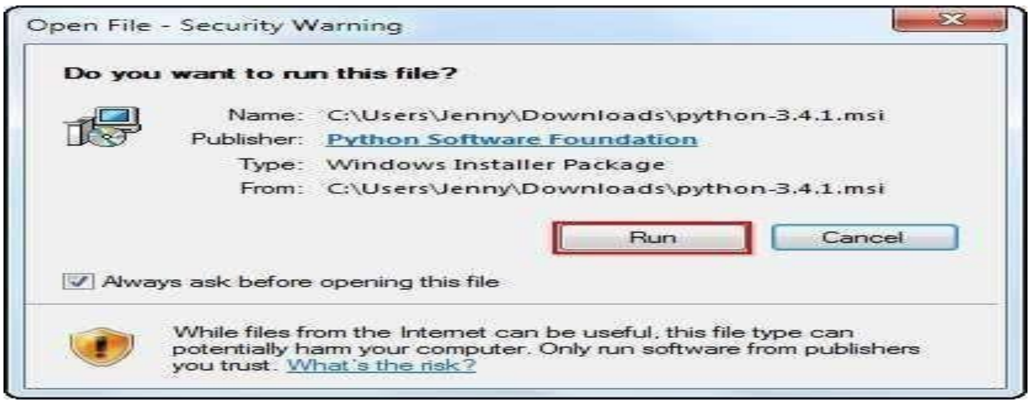
Now that you are on the download page, select which of the software builds you would like to download. For the purposes of this article we will use the most up to date version available (Python 3.10.5).

Once you have clicked on that you will be taken to a page with a description of all the new updates and features of 3.10.5,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 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.

2.2 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.


Open File - Security Warning

Do you want to run this file?

Name: C:\Users\Jenny\Downloads\python-3.4.1.msi
Publisher: Python Software Foundation
Type: Windows Installer Package
From: C:\Users\Jenny\Downloads\python-3.4.1.msi

Run | Cancel

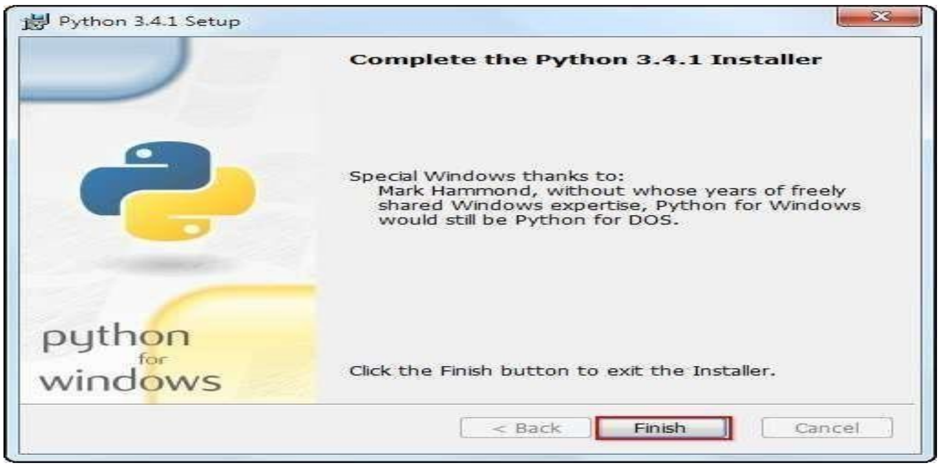
[✓] Always ask before opening this file

While files from the Internet can be useful, this file type can potentially harm your computer. Only run software from publishers you trust. What's the risk?

Fig:2.2: run python software

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."

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


Python 3.4.1 Setup

Complete the Python 3.4.1 Installer

Special Windows thanks to:
Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

Click the Finish button to exit the Installer.

< Back | Finish | Cancel

Fig:2.3.completion of installation

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: “Python path.” The string that you will need to enter is: “C:\Python27\;C:\Python27\Scripts;”

2.3 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.

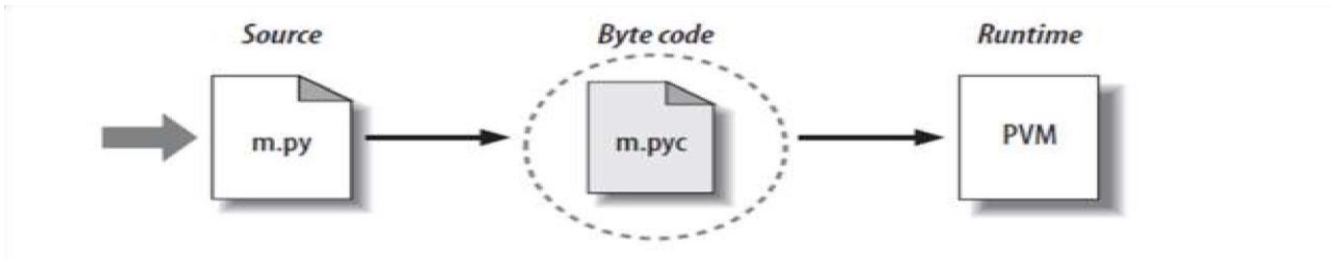
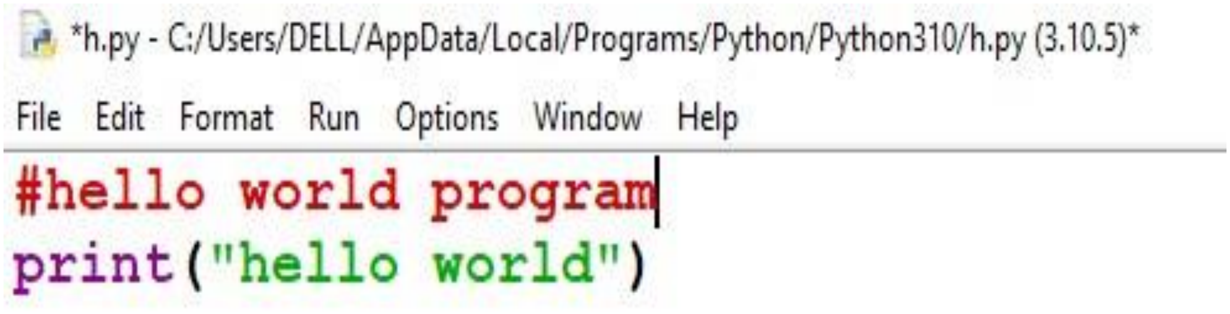


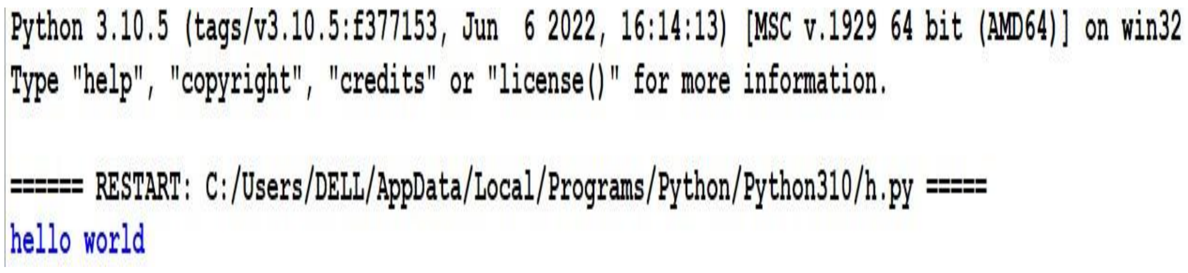
Fig:2.4 Diagram of execution

Eg:


h.py - C:/Users/DELL/AppData/Local/Programs/Python/Python310/h.py (3.10.5)
File Edit Format Run Options Window Help

```
#hello world program
print("hello world")
```

Fig:2.5. Code for hello program



```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

====== RESTART: C:/Users/DELL/AppData/Local/Programs/Python/Python310/h.py =====
hello world
```

Fig:2.6. result of hello program

CHAPTER 3

BASICS OF PYTHON

3.1 Tokens:-

The Python Interpreter uses lexical analysis to convert source code into the machine code. It is the lowest level of syntax checking . In this analysis the program is divided into many small pieces called tokens. The token is nothing but unit of processing.

There are following token in python:

- **Keywords**
- **Identifiers**
- **Literals**
- **Operators**

3.1.1 Keywords:-

Keywords are special reserved words which convey a special meaning to the compiler/interpreter. Each keyword have a special meaning and a specific operation. List of keywords used in python are:

True	False	None	And	As
Asset	Def	Class	Continue	Break
Else	Finally	Elif	Del	Except
Global	For	If	From	Import
Raise	Try	Or	Return	Pass

Nonlocal	In	Not	Is	Lambda

Table : 3.1: keyword in python

3.1.2 Identifiers:-

Identifiers are the names given to the fundamental building block in a program. These can be variables, class, object, functions, lists, dictionaries ,etc.

An Identifiers is a long sequence of characters and numbers.

No special character except underscore (_) can be used as an identifiers.

Keyword should not be used as an identifiers name.

Python is case-sensitive, So using case is significant.

First character of an identifiers can be character, underscore(_) but not digit.

3.1.3 Operators :-

There are following operator:

1. Arithmetic Operator
2. Relational Operator
3. Assignment Operator
4. Logical Operator
5. Membership Operator
6. Identity Operator
7. Bitwise Operator

Arithmetic operators:-

//	integer division
+	To perform addition
-	To perform subtraction
/	To perform division
%	To return remainder after division
**	Perform exponent

Relational operator:-

<	Less than
>	Greater then
=<	Less than or equal to
=>	Greater than or Equal to
==	Equal to
!=	not Equal to

Assignment operator:-

=	Assignment
/=	Divide and assign
-=	Subtract and assign
*=	Multiply and assign
%=	modulus and assign
**=	Exponent and assign

Logical operators:

And	Logical AND(when both cond. are true output will be true).
Or	Logical OR(if any one condition is true output will be true).

Membership operator:-

In	return true if variable in sequence of another variable, else false
Not in	return true if variable is not in sequence of another variable, else false.

Identity operators:- is return true if identity of two operands are same, else false. Is not return true if identity of two operand are not same, else false.

- **Operation of arithmetic operators:**
- **Integer division-**The quotient returned by this operator is dependent on the argument being passed.
- **Floating point division-**The quotient returns by this operator is always a float number, no matter if two numbers are integer
- **Remainder-** modulo operator (%), which returns the remainder of dividing two numbers.
- **Exponent-**The operator that can be used to perform the exponent arithmetic in Python is **.
-

3.2 Selective Statements:-

The if statements in python is same as C language which is used test condition.

If condition is true statement of if block is executed otherwise it is skipped.

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).

Table:3.2 : if else statement

3.3 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.

There are two types of loops:

While loop and For loop

3.3.1 While Loop:-

It is used to execute number of statements or body till the condition passed in while is true.

Once the condition is false, the control will come out of the loop.

```
Syntax:   While<expression>
                    body
```

Example:

```
A=10

While a>0:

        Print("value of  a is ", a)

        a=a-2
```

3.3.2 For Loop-

It is used to iterate a variable over a sequence in the order than they appear.

Syntax:

```
                for <variable> in <sequence>:

                        body
```

Eg:

```
For i in range(1,5)

   Print i
```

O/P:-

1 2 3 4 5

3.4 Python Comments:-

1)**single line comments()** : in case user wants to specify a single line comments, then comment start with #.

Syntax: **# this is simple line comment**

2) **multi line comment()** : multi lined comment can be given inside triple quotes.

Syntax: “””this

Is

the

Multiple comments”””

3.5 Data Types :-

Python provides various standard data types that defines the storage method on each of them**.** It represents the kind of value that tells what operations can be performed on a particular data. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classe**s.** There are following types of data types:-

3.5.1 List:-

- Python list are the data structure that is capable of holding different types of data.
- Python list are mutable i.e., Python will not create a new list if we modify an elements in the list.
- It is a container that holds other object in the given order. Different operation like insertion and deletion can be performed on lists.
- A python list is enclosed between square([]) brackets.

Syntax:

<list_name>=[value 1, value 2, value 3, ….,, value n]

List function and methods:-

SN	Function with Description
1	Cmp(list1,list2)- Compares elements of both lists.

2	Len(list1)- gives the total length of the list.
3	Max(list1,list2) Returns item from the list with max value.
4	Min(list1,list2)- Returns item from the list with min value.
5	List(seq1)-converts a tuple into list.

Table:3.3:List function

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 function if given

Table:3.4: List methods

3.5.2 Tuple:-

- A tuple is a sequence of immutable objects, therefore tuple cannot be changed.
- The object are enclosed within parenthesis and separated by comma.
- Tuple is similar to list. Only the difference is that list is enclosed within square brackets ,tuple within parenthesis and List have mutable objects whereas tuple have immutable work.
- Tuple can be access in the same way as list.

Eg:

```
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]
```

Result-

```
tup1[0]:

physics

tup2[1:5]

[2, 3, 4, 5]
```

3.5.3 Dictionary:

- Dictionary is an unordered set of key and value pair.
- It is an container that contains data, enclosed within curly braces.
- The pair i.e., key and value is known as itm.
- The key and value is separated by a colon (:). This pair is known as item. Item are separated from each other by a comma(,). Different items are enclosed within curly braces and this form dictionary.
- Dictionary can be mutable i.e., value can be updated.
- Key must be unique and immutable. Value is accessed by a key. Value can be updated while key cannot be changed.
- Dictionary is known as Associative array since Key works as index and they are decided by the user.
- The key passed in the item must be unique.

Example :

```
data ={100:’Ravi’,101:’vijay’,102:’rahul’]

print(data)
```

o/p

{100: ‘ravi’ , 102: ‘vijay’, 102: ’rahul’}

Function of dictionary:

Method	**Description**
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
keys()	Returns a list containing the dictionary's keys
Setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Table : 3.5: Dictionary function

Eg;

#key   print("keys:")

data={101:'rahul',102:'alok',103:'suraj'} print(data.keys())

```
#values  print("values:")

data2={101:'rahul',102:'alok',103:'suraj'}

print(data2.values())
```

3.5.4 Sets:-

A set is a collection of items not in any particular order. A python set is similar to this mathematical definition with below additional conditions.

- The elements in the sets cannot be duplicates. A={2,3,5,7,7}

  ```
   print(a)
  o/p
  {2,3,5,7}
  ```

- The elements in the set are immutable(cannot be modified).but set as a whole is mutable.
- There is no index attached to any elements in python set. So they do not support any indexing or slicing operation.

Eg:

```
a={“a”,”b”,”c”}
 print(a)
 output:
 {‘a’,’b’,’c’}
```

Using construction set():

```
b=set((“a”,”b”,”c”))

print(b)

c=set((“aa”,”bb”,”cc”))

print(c)

 output:

{‘a’,’b’,’c’}     {‘aa’,’bb’,’cc’}
```

Set Operation:-

- **Union – use operator -** The union of two sets is the set of all the elements of both the sets without duplicates
- **Intersection and minus-** The intersection of two sets is the set of all the common elements of both the sets.

3.5.5 String:-

- String are simplest and easy to use in python.
- String python are immutable.
- We can simply create python string by enclosing a text in a single as well as double quotes.
- In python, String are stored individual characters in a contiguous memory location.
- The benefits of using string is that it can be accessed by from both the direction in forward and backward.
- Both forward as well as backward indexing are provided using string in python.

- **String Operators:**

  There is basically 3 types of operators by string:

  - Basic operators
  - Membership Operators
  - Relational Operators.

- **Basic operators:**

There are two types of basic operators in string. They are “+” and “*“.

Example:

‘10’+’50’ = ‘1050’

‘A’+’1001’ = ‘A1001’

“VGT” * 2 = “VGTVGT”

- **Membership operators:**

There are two types of membership operators:

1) The “In” operators return true if a character or the entire substring is present in the specified string, otherwise false
2) The “not in” operators return true if a character or entire substring does not exits in the specified string, otherwise false.

- **Relational Operators:**

All the comparison operator(<,>,<=,>=,==,<>) are also capable to string . The string are compared based on the ASCII value or Unicode.

Eg:

“RAJAT”==”RAJAT”	TRUE
“afsha”>=”Afsha”	TRUE

String Methods:

Function Name	Description
capitalize()	Converts the first character of the string to a capital (uppercase) letter
isalnum()	Checks whether all the characters in a given string is alphanumeric or not
isalpha()	Returns “True” if all characters in the string are alphabets
isdecimal()	Returns true if all characters in a string are decimal
isdigit()	Returns “True” if all characters in the string are digits
find()	Returns the lowest index of the substring if it is found
count()	Returns the number of occurrences of a substring in the string.
upper()	Converts all lowercase characters in a string into uppercase

Table :3.6: String methods

CHAPTER 4

FUNCTION

4.1 Function:-

- A function is self-block of code.
- A function can be called as a section of a program that is written once and can be executed whenever required in the program, thus making code reusability.
- A function is a subprogram that works on data and produce some output.
- 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.

Syntax:

```
def <function_name> ([parrameters]);
        body
```

4.1.1 Default Arguments:-

Default arguments is the argument which provides the default values to the Parameter passed in the function definition, in case value is not provided in the function call.Default values must be specified from right side of list of arguments of function.

4.1.2 Keyword Arguments:-

Using the keyword arguments, the arguments passed in function call is matched with function definition on the basis of name of the parameter.

4.2 Scope of variable :-

Scope of variable can be determined by the part of which variable is defined. Each variable cannot be accessed in each part of program. There are two types of variables based on scope:

1) Local Variable
2) Global variable

4.2.1 Local variable:

Variable declared inside a function is known as Local variable. These have a local access thus these variable cannot be accessed outside the function body in which they are declared.

4.2.2 Global variable:

Variable defined outside the function is called global variable. Global variable is accessed all over program thus global variable have widest accessibility.

Eg:

```
 A=20

Def msg: a=10

        print("value of a is:",a)

        return msg()

print(a)
```

```
o/p value of a is 10
20
```

4.3 Anonymous Function :-

- Anonymous function are the function that are not bond to me.
- Anonymous functions are created by using a keyword “lambda”.
- Lambda takes any number of arguments and return an evaluated expression.
- lambda is created without using def keyword.

An anonymous function is a function that is defined without a name. While normal functions are defined using the def keyword in Python, anonymous functions are defined using the lambda keyword. Hence, anonymous functions are also called lambda functions.

Eg:

```
Square=lambda a:a*a
 print(“Square of number is:”,square(10))
o/p-
 100
```

CHAPTER 5

MODULE AND PACKAGE

5.1 Module:-

A module is simply a file where classes, functions and variable are defined. Group similar code into a single file makes it easy too access. It is the files which have similar code. This module is simplify a python code where classes, variables and function are defined.

Advantages:

1) Reusability : module can be used in some other python code. Hence it provides the Facility of code reusability**.**
2) Categorization: Similar types of attributes can be placed in one module.

Eg:

```
Def add(a,b) :
                 c=a+b
                print(c)
                return def subtract(a,b) :
                     c=a-b print(c)
                    return
```

File **mymath.py** has following functions

```
import mymath
a=mymath.add(10,20)
b=mymath.subtract(30,10)
print(a,b)
```

5.1.1 The import Statement:-

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

A module is loaded only once, regardless of the number of times it is imported. This prevents the module execution from happening over and over again if multiple imports occur.

We can use import in many ways:

- From modulename import * To import all the function of specified module.
- From modulename import fun1, fun2…. To import only specified function of a module.

5.2 Package:-

A package is simply collection of similar module, subpackages. Modules that are related to each other are mainly put in the same package. When a module from an external package is required in a program, that package can be imported and its modules can be put to use.

Steps to create and import package:

1) Create a directory, say mymath.
2) Place different modules inside the directory. We are placing 2 modules advance.py and basic.py

Advance.py

```
def poewer(x,y) :
if (y==0) :

                return 1
else:

                return x*power(x,y-1)
```

basic.py

```
def

    add(a,b) :
          c=a+b

     return c
```

1) Create a file __init__.py in folder mymath and write following code.
2) Import mymath.basic
3) Import mymath.advance
4) Use function of modules in your python program, which is not in folder mymath

Syntax:

```
import  mymath
a=mymath.basic.add(2,3)
print(a)
```

for execution from mymath.basic

```
import add a=add(2,3)
print(a)
```

5.3 Math Function:-

The math module is a standard module in python and is always available. To use the mathematical function under this module, you have to import the module using import math. A math module in python in simple terms is a group of python(.py) code files that can be imported into some other python program. In simple terms, consider a module as a library that has some prewritten code that could be reused in your code.

List of some function in python math module

Factorial()	Return the factorial of x
Exp()-	return e**x
Log2(x)	return the base-2 logarithm of x
Pow(x,y)	return x raised to the power of y
Sqrt()	return the square root of x
Pi()	mathematical constant, the ratio o
Sqrt()	return the square root of x
Pi()	mathematical constant, the ratio o
circle to its	f circumference of a
e()	diameter(3.141)
log10(x)	mathematical constant e(2.718….)
math.dist()	return the base base-10 algorithm of x.
(p and q),	Returns the Euclidean distance between two points

Table 5.1: Math function

<u>Eg:</u>

```
#square root
Import math
a=math.sqrt(4)
 b=math.pow(2,3)
 c=math.factorial(4)
print(a)
print(b)
print(c)
```

o/p:-

```
2.0
8.0
24
```

CHAPTER 6

PYTHON EXCEPTION HANDLING

6.1 Exception Handling:-

Exception is run time error, which occur at execution of program. Whenever an exception occurs the program stops automatically. To stop termination of program we use exception handling. Exception is the base class for all the exceptions in Python.

Exceptions are raised when the program is syntactically correct, but the code resulted in an error. This error does not stop the execution of the program, however, it changes the normal flow of the program.

6.1.1 Handling an exception:-

If you have some suspicious code that may raise an exception, you can defend your program by placing the suspicious code in a try: block.

After the try: block, include an except: statement, followed by a block of code which handles the problem as elegantly as possible.

6.1.2 Common Exception:-

- ZeroDivisionError : Occurs when a number is divided by zero.
- NameError: It occurs when a name is found. It may be logical or global.
- IndentionError: If incorrect identation is given.
- IOError: It occurs when input output operation fails.
- EOFError: It occurs when end of the file is reached and yet operation are being performed.

Syntax:

- try:

```
    malicious code except
Exception 1:
    Execute code except
Exception 2:
    Execute code
Except Exception N:
    Execute code
else:
    In case of no exception , execute the else block code.
```

Note: python allows us to declare exceptions using the same except statement.

6.2 Finally Block:-

In case if there is any code which the user went to be executed, whether exception occurs or not then that code can be placed inside finally block. Finally block will always be executed irrespective of the exception.

Synta
x:

```
try:
code
 finally:
   code which is must to be executed.
```

6.3 Raise Exception:-

Raise will cause an exception to occur and thus execution control will stop in case, it is not handled. The sole argument to raise indicates the exception to be raised. This must be either an exception instance or an exception class (a class that derives from Exceptions)

In some situations, you might want to run a certain block of code if the code block inside ‘try’ ran without any errors. For these cases, you can use the optional ‘else’ keyword with the ’try’ statement.

Eg:

```
try:

num = int(input("Enter a number: "))
 assert(num%2==0) except:
   print("Not an even number!")
 else:
   reciprocal = 1/num
print(reciprocal)
```

output:

```
Enter a number: 1

Not an even numbe
```

OBJECT ORIENTED PROGRAMMING

7.1 OOP :-

Python is an object oriented programming language. It allows us to develop application using object oriented approach. In python, we can easily create and use classes and objects.

The major principles of object oriented programming system are given below :

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

7.2 Class:

Class can be defined as a collection of objects. It is a logical entity that has some specific attributes and methods.

For ex: if you have an employee class then it should contain an attribute and methods.

Syntax:

class ClassName:

<statement 1> ….. <statement 2>

7.3 Inheritance:-

Inheritance is a feature of object oriented programming. It is used to specify that one class will get most or all of its feature from its parent class. It is very powerful feature which facilitates users to create a new class with a few or more modification to an existing class. The new class is called child class or derived class and the main class from which it inherits the properties is called base class or parent class.

The child class or derived class inherits the feature from the parent class, adding new feature to it. It facilitates re-usability of code.

Syntax:-

```
        class DerivedClassName(BaseClassName):

                <statement 1>

                …

                …

                …

                <statement 2>
```

7.3.1 Overriding Methods:-

You can always override your parent class methods. One reason for overriding parent's methods is because you may want special or different functionality in your subclass.

Syntax:-

```
Class parent:

 def myMethod(self):

print 'Calling parent method'

 class Child(Parent): # define child class
```

```
def myMethod(self):

print 'Calling child method c = Child() # instance of child

 c.myMethod()'
```

7.4 Constructor:-

A constructor is a special type of method(function) which is used to initialize the instance members of the class. Constructor can be parameterized and non parameterized as well. Constructor definition executes when we create objects of class. Constructor also verify that there are enough resources for the object to perform any start-up task.

Creating a constructor :-

A constructor is a class function that begin with double uunderscore(_). The name of constructor is always the same __init__().

Eg:

```
class Student:

            def __init__(self):

            print("this is non parameterized constructor")

             def show(self,name):

                        print("hello",name)

                         s1=Student() s1.show("GRRAS")

 output:

 this is non parameterized constructor hello GRRAS
```

7.4.1 Parameterized constructor:-

When we declare a constructor in such a way that it accepts the arguments during object creation then such type of constructors are known as Parameterized constructors. As you can see that with such type of constructors we can pass the values (data) during object creation, which is used by the constructor to initialize the instance members of that object.

```
  class student:
  def __init__(self,name,branch):
                 self.name=name
                 self.branch=branch
 def display(self):
        print("name:"+ self.name+" \nbranch:"+ self.branch)
 s1=student("A","CS")
s2=student("B","EC")
s1.display()
s2.display()
```

output:

```
  name:A
  branch:CS
 name:B
branch:EC
```

CHAPTER 8

FILE HANDLING

8.1 File Handling:-

Python provides the facility of working on files. A file is an external storage on hard disk from where data can be stored and retrieved.

Step1-

open the file using file object = open(file_name [, access_ mode] [,buffering]) Modes:

R	it open in reading modes. It is the default modes of file. Pointer is at beginning of the file.
W	open files in writing mode. If file already exists, then overwrite the file else create a new file.
A	opens files in appending modes. If file already exists, then append the data at the end of existing file, else create new file.

Table :8.1: access mode

Step 2:-

Read or write operation

Writefileobject.write(string str) fileobject.read(value)

Step 3:-

choose the file

Fileobject.close()

Eg:

#Read text from file

```
Obj1=open("abc.txt","r")
s1=obj1.read(20) print(s1)
obj1.close()
```

#Save text to file

```
Obj=open("abcd.txt","w")

Obj.write("welcome to the world of python")

Obj.close()
```

File attributes-

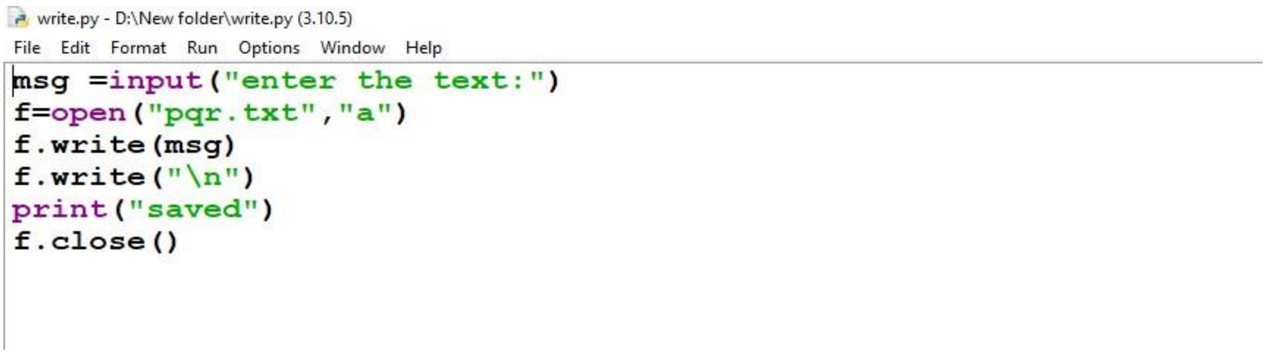
Name	Return the name of file
Mode	Return the mode in which file is being opened.
Closed	Return boolean value. True, in case if file is closed else false

Function of file handling:-

- Rename()- It is used to rename a file. It takes two arguments, existing_file_name and new_file_name.
- Syntax: os.rename(exixting_file_name, new_file_name)
- Remove()- It is used to delete a file. It takes one arguments. Pass the name of the file which is to be deleted as the argument of method.
- Syntax: os(file_name)
- Mkdir()- It is used to create a directory. A directory contains the files. It takes one argument which is the name of the directory.
- Syntax: os.mkdir("file_name")

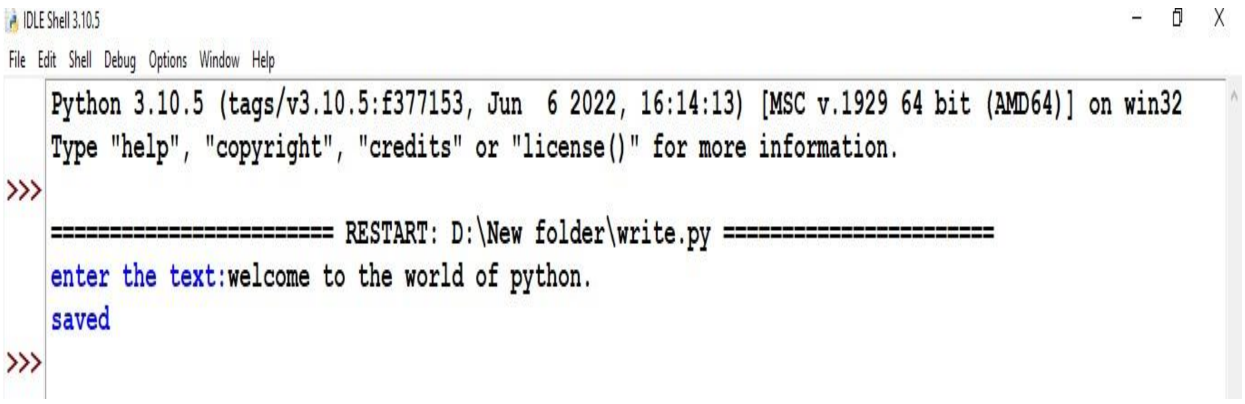
- Rmdir()- It is used to delete a directory. It takes one arguments which is the name of the directory
- Syntax: os.rmdir("directory_name)
- Chdir()- It takes used to changed the current working directory.it takes one arguments which is the name of the directory.
- Syntax: os.chdir("file_name")

Eg:


write.py - D:\New folder\write.py (3.10.5)
File Edit Format Run Options Window Help

```
msg =input("enter the text:")
f=open("pqr.txt","a")
f.write(msg)
f.write("\n")
print("saved")
f.close()
```

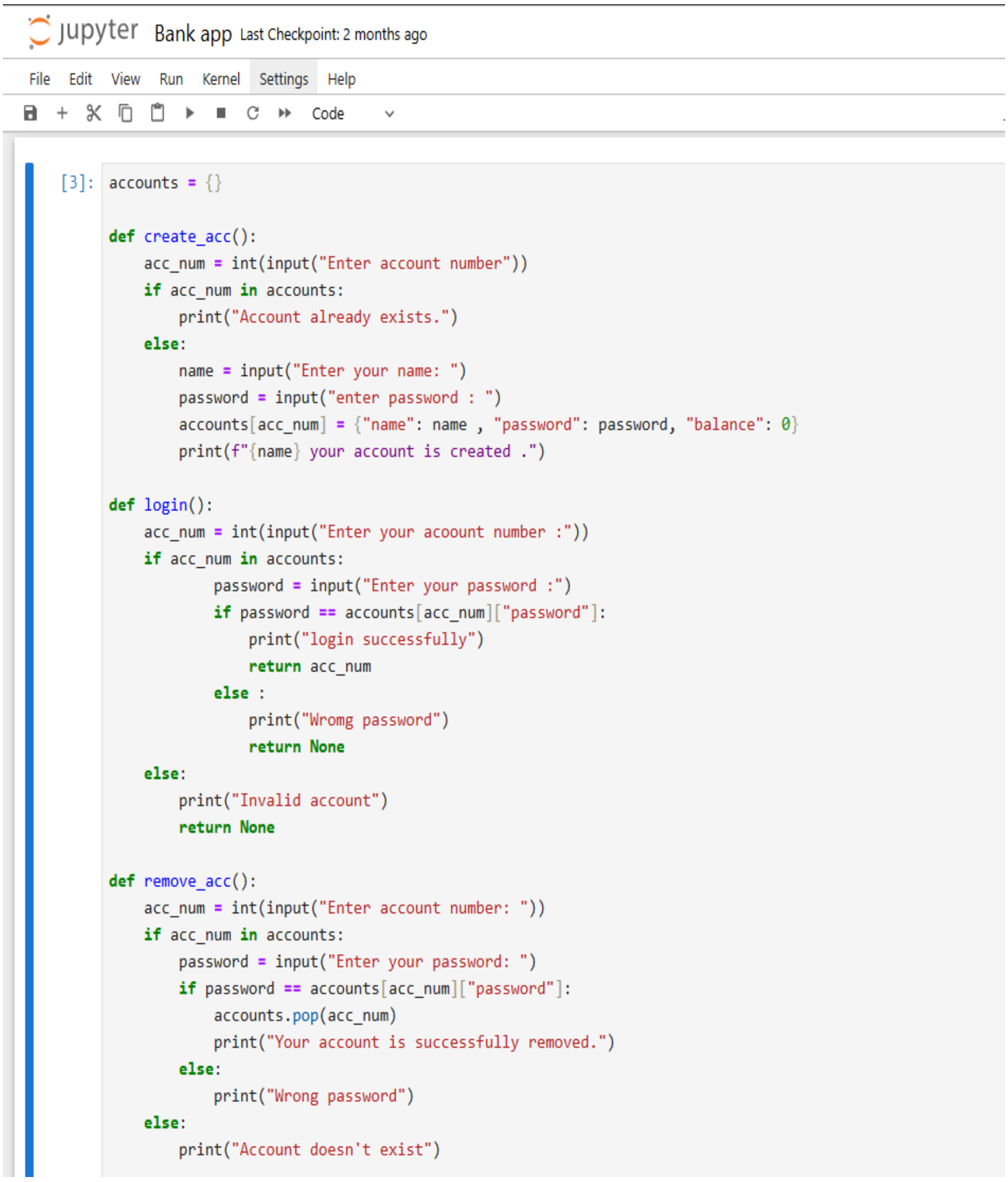
Fig:8.1:file handling Example



```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======================= RESTART: D:\New folder\write.py ======================
enter the text:welcome to the world of python.
saved
>>>
```

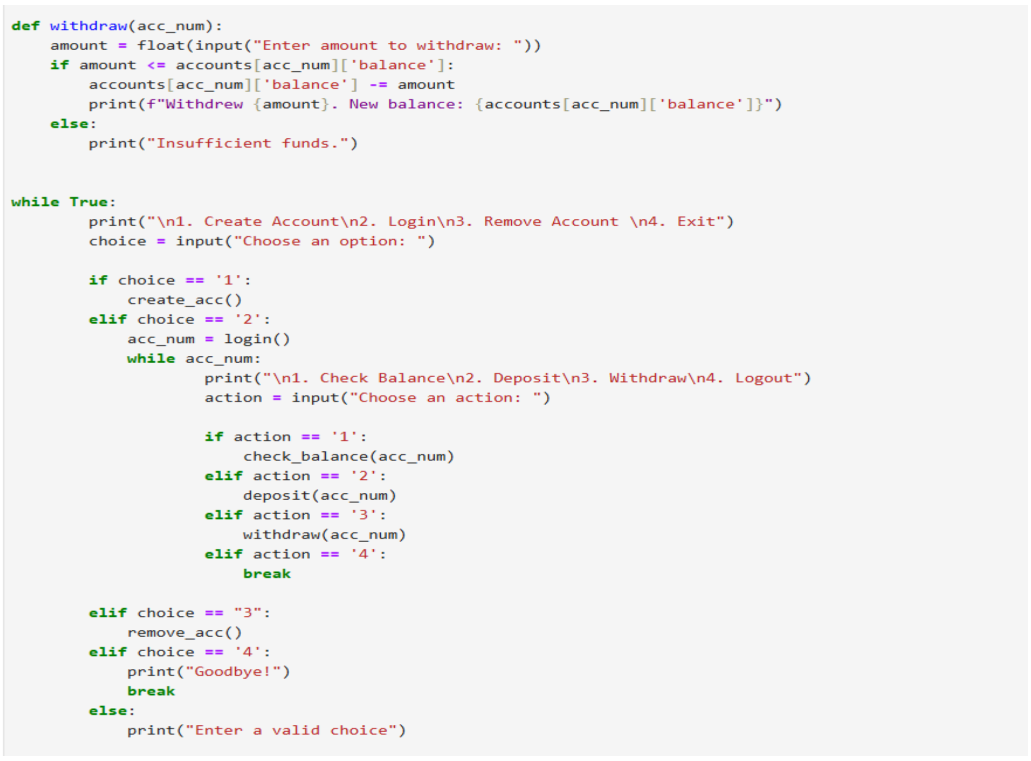
Fig:8.2 Result of example

PROJECT : BANK APP


jupyter Bank app Last Checkpoint: 2 months ago
File Edit View Run Kernel Settings Help
Code

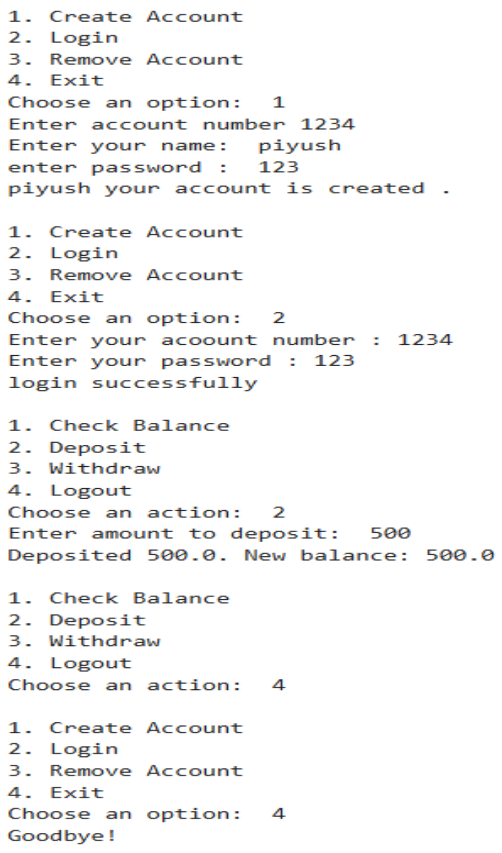
```
[3]: accounts = {}

def create_acc():
    acc_num = int(input("Enter account number"))
    if acc_num in accounts:
        print("Account already exists.")
    else:
        name = input("Enter your name: ")
        password = input("enter password : ")
        accounts[acc_num] = {"name": name , "password": password, "balance": 0}
        print(f"{name} your account is created .")

def login():
    acc_num = int(input("Enter your acoount number :"))
    if acc_num in accounts:
            password = input("Enter your password :")
            if password == accounts[acc_num]["password"]:
                print("login successfully")
                return acc_num
            else :
                print("Wromg password")
                return None
    else:
        print("Invalid account")
        return None

def remove_acc():
    acc_num = int(input("Enter account number: "))
    if acc_num in accounts:
        password = input("Enter your password: ")
        if password == accounts[acc_num]["password"]:
            accounts.pop(acc_num)
            print("Your account is successfully removed.")
        else:
            print("Wrong password")
    else:
        print("Account doesn't exist")
```

Fig.9.1.code(a)



```python
def withdraw(acc_num):
    amount = float(input("Enter amount to withdraw: "))
    if amount <= accounts[acc_num]['balance']:
        accounts[acc_num]['balance'] -= amount
        print(f"Withdrew {amount}. New balance: {accounts[acc_num]['balance']}")
    else:
        print("Insufficient funds.")


while True:
        print("\n1. Create Account\n2. Login\n3. Remove Account \n4. Exit")
        choice = input("Choose an option: ")

        if choice == '1':
            create_acc()
        elif choice == '2':
            acc_num = login()
            while acc_num:
                    print("\n1. Check Balance\n2. Deposit\n3. Withdraw\n4. Logout")
                    action = input("Choose an action: ")

                    if action == '1':
                        check_balance(acc_num)
                    elif action == '2':
                        deposit(acc_num)
                    elif action == '3':
                        withdraw(acc_num)
                    elif action == '4':
                        break

        elif choice == "3":
            remove_acc()
        elif choice == '4':
            print("Goodbye!")
            break
        else:
            print("Enter a valid choice")
```

Fig.9.2.code(b)



```
1. Create Account
2. Login
3. Remove Account
4. Exit
Choose an option:  1
Enter account number 1234
Enter your name:  piyush
enter password :  123
piyush your account is created .

1. Create Account
2. Login
3. Remove Account
4. Exit
Choose an option:  2
Enter your acoount number : 1234
Enter your password : 123
login successfully

1. Check Balance
2. Deposit
3. Withdraw
4. Logout
Choose an action:  2
Enter amount to deposit:  500
Deposited 500.0. New balance: 500.0

1. Check Balance
2. Deposit
3. Withdraw
4. Logout
Choose an action:  4

1. Create Account
2. Login
3. Remove Account
4. Exit
Choose an option:  4
Goodbye!
```

Fig.9.3.Bank app

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.
- With a vast collection of libraries, Python allows students to explore fields like data science, machine learning, web development, and more without needing additional programming languages. This versatility also means students can see practical applications of their skills early on.
- Python is cross-platform, meaning code written on one operating system (Windows, macOS, or Linux) can typically run on another with little to no modification. This flexibility allows students to collaborate and work on various devices, fostering a more inclusive learning environment.

FUTURE SCOPE

The future of Python looks extremely promising due to several factors. Here are a few key areas where Python is expected to continue making significant strides:

1. Artificial Intelligence and Machine Learning

Python has established itself as the go-to language for AI and ML, thanks to its simplicity and the powerful libraries available, such as TensorFlow, Keras, and Scikit-learn. As these fields grow, Python's role in them will likely expand, driving innovation in technology and various industries.

2. Data Science and Analytics

With the exponential growth of data, the need for efficient data analysis tools is greater than ever. Python, with libraries like Pandas, NumPy, and Matplotlib, is a favorite among data scientists for its ability to handle complex data sets and perform data visualization. The demand for data scientists proficient in Python will continue to rise.

3. Web Development

Frameworks like Django and Flask have made Python a popular choice for web development. The simplicity of the language and the robustness of these frameworks ensure that Python remains a preferred language for building scalable and secure web applications.

4. Automation

Python's ease of use and powerful scripting capabilities make it an excellent choice for automation tasks. From simple scripts to complex workflows, Python is used to automate repetitive tasks across various domains, including network automation, testing, and data processing

5. IoT (Internet of Things)

Python's role in IoT is growing as well, with libraries like MicroPython and frameworks that support hardware interaction.

REFERENCE

- http://python.org
- https://stackoverflow.blog/2021/07/14/getting-started-with-python
- https://en.wikipedia.org/wiki/python_(programming_language)
- https://www.geeksforgeeks.org/
- https://www.vgtindia.org/training/training.python
- https://grras.com/page/GRRAS-Solutions-Pvt-Ltd