**Industrial Training Report**

On

***PYTHON PRO PROGRAMMING***

submitted in partial fulfilment For the award of the degree of

***Bachelor of Technology***

***In computer engineering***



Submitted To: Submitted By:

MR. Pawan Sen Varun Singh

HOD CSE 5th Sem CSE

22EAICS952

Department of computer science & Engineering

Arya institute of Engineering & Technology Jaipur

Rajasthan Technical University Kota (2023-2024)

ARYA INSTITUTE OF ENGINEERING & TECHNOLOGY

SP-40, RIICO Industrial Area Jaipur (Raj)-302028

Department of Computer Science & Engineering

***CERTIFICATE***

*This is to certify that the work ,which is being is presented in the Practical training seminar report for* **“PYTHON PRO ROGRAMMING”** *Submitted by* **Mr. Varun Singh** *student of third year (5th Sem)B-tech in computer science as a partial fulfillment for the award of degree of bachelor of technology is a record of student work Carried out and found satisfaction for submission.*

Mr. …………………….. ……………………………….

Training Coordinator Head of Department



***Candidate’s Declaration***

I hereby declare that the work, which is being presented in the Industrial Training report, entitled “***Python Pro Programming***” in partial fulfillment for the award of Degree of “***Bachelor of Technology***” in Department of Computer Science & Engineering with Specialization in Computer Engineering and submitted to the Department of Computer Science & Engineering, ***Arya College of Engineering***, is a record of my own investigations carried under the Guidance of ***Mr. Sachin Sharma***, Assistant Professor, Department of Computer Science & Engineering.

**(Signature of Candidate)**

**Candidate Name**

**Varun Singh**

**Roll No: 22EAICS952**

***Abstract***

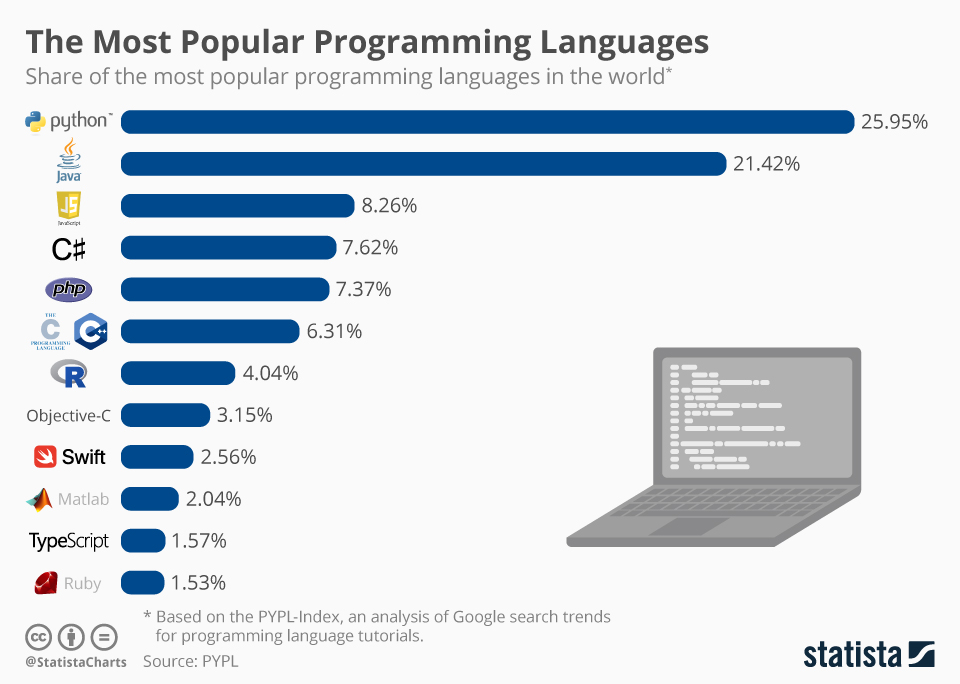
Python is a high-level, interpreted programming language that emphasizes code readability and simplicity. Created by Guido van Rossum and first released in 1991, Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming. It is widely used in various domains, such as web development, data science, artificial intelligence, machine learning, automation, and scientific computing, owing to its large ecosystem of libraries and frameworks.

Python's design philosophy prioritizes ease of learning and use, with an emphasis on reducing the complexity of code, making it ideal for both beginners and professionals. Python also supports dynamic typing and garbage collection, which simplifies memory management. Its versatility allows developers to use Python for small scripts as well as large-scale applications.

One of Python’s key strengths is its extensive standard library, which provides a rich set of tools for handling common tasks such as file I/O, data manipulation, web services, and database access. Additionally, Python's cross-platform capabilities enable it to run on various operating systems such as Windows, macOS, and Linux.

Python has become an integral part of various modern technologies and industries, from powering web frameworks like Django and Flask to facilitating data analytics with libraries such as NumPy, Pandas, and TensorFlow. Its growing popularity is also driven by an active community that continuously contributes to its

development, ensuring that Python remains relevant in an evolving technological landscape.

****

***Acknowledgement***

It is our proud and duty to acknowledgement the kind of help and guidance received from several people in prepare this report in this form without their valuable help ,cooperation and guidance

First and foremost ,we wish to record our sincere gratitude to the teacher for his constant support and encouragement in preparation of this report and for making available library and laboratory facilities needed to prepare this report.

The seminar on **”Python Pro PRogramming”** was very helpful to us in given the necessary background information and inspiration in choosing this topic for seminar .their contribution and technical support in preparing this report are greatly acknowledged.

Let but not the we wish to thank our parents for financing our studies in this college as well as for constantly encouraging us to learn engineering us to learn engineering .their perposal sacrificiaI providing this opportunity to learn engineering is gratefully acknowledgement

***Table Of Content***

**# Introduction**

* Python
* Object Oriented Programming Language (OOP)
* History

**# Data Types & operator**

* Data types
* Variable
* String
* operator

**# Tuple & List**

* Tuple
* List

**# Inheritance**

* Inheritance
* Property of Inheritance

**# Polymorphism**

* Polymorphism
* Types of polymorphism

**# Loop**

* For
* While
* Nested

**# Project**

**Bank Management System**

***Python***

Python is a widely used high-level general –purpose interupted dynamic prog language .its design philosophy emphasized code Redability ,and its syntax allow prog to express concept in fewer lines of code than would be possible in language such as C++ or java. The language provide construct intended to enable clear prog on both a small and large scale.

Python support multiple prog paradigms , including object oriented ,imprative and functional prog or procedural .it feature a dynamic type system and automatic memory management and has a large and comprehensive standard library.

***Object oriented programming language***

Object-oriented (oop) is a prog paradigm based on the concept of “object”,where may contain data in the form of field,often know as attributes ; and code ,in the form of procedure ,often know as method . A distinguished feature of object is that an object procedure can acces and modify the data field of the of the object

with which they are associated (object have a notion of “this “or ”self”)

In OOP computer programs are designed by making them out of object that interact with one another. There is significant diversity in object oriented programs, but most popular language are class-based, meaning that object are instances of classes which typically also determines their types.

***History***

Python was coceived in late 1980s, and its implementation was started in december 1989 by ***Guido Van Rossum*** at the Netherland as a successor to the ABC language (itself inspired by SETL) capable of exception handling and interface with the Amoeba operating system . Van Rossum is Python’s principle author.



***Guido Van Rossum***

***Data Type***

Data type determine whether an object can do something, or whether it just would not make sense .other prog lang often determine wheather an operation make sense for an object by making sure the object can never be stored some where the operation will be perfomed on the object (this type system is called static typing )

***Python has many native data types:***

**Boolean:** are either true or false.

**Number:** can integer (1& 2), floats(1.1 &1.2 **),Fraction (1/2 & 2/3),** Or even complex number.

**String:** are sequences of Unicode character ,eg an HTML doc.

**Bytes and byte array:** eg. a JPEG image file

**List:** are orderd sequences of value

**Tuple:** are orderd , immutable sequences of value.

**Set:** are unordered bags of value

***Variable***

Variable are nothing but reserved memory location to store value this mean 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 variable .

Ex: counter=100 # An integer

Assignment miles =100.0 # A floating

Point name = ”John” # A string

***String***

In programming term , we usually call text a string .when you think of as a collection of letter , the term make sense.

All the letter ,number and symbol in this book could be a string for that matter , your name could be astring ,and so could your address.

***Creating string***

In python ,we create a string by quates(“ ”, ‘ ‘ ) around text . for example ,we could take our otherwise useless.

“hello”+”World” “helloworld” #concatenation

“hello”\*3 “hellohellohello” #repetition

|  |
| --- |
| **Operator** **Example** |
| **+**  x+y+2 |
| **-** x-y-2 |
| * x\*y |
| / x/y |
| % x%y |
| \*\* x\*\*y |

***Tuples***

A tuple is a sequences of immutable Python objects. Tuples are sequences ,just like lists . the difference b/w tuples and are ,the tuples cannot be changed unlike list and and tuples use parentheses.

**Accesing Value in Tuples:**

To access value in tuple use the square braket for slicing along with the index or indices to obtain value available at that index.

for example:

tup1=(‘physic’,’chemistry’,’1997,2000);

tup2 =(1,2,3,4,5,6,7);

print “tup1[0]: “

tup1[0]

print “tup2[1:5]:”

Output = tup1[0]:

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

***List***

The list is a most versatile data type available in python which can be written as a list of comma – seprated value (item) b/w square bracket .important thing about a list is that item in a list need not be of the same type.

Creating a list is as putting different comma-seprated value b/w square bracket .

**Example=** list1=[‘physic’,chemistry’,1997,2000];

list2=[‘1,2,3,4,5];

list3=[“a”,”b”,”c”]

**Accessing value in lists:**

To accessing value in list ,use the square bracket for slicing along with the index or indices to obtain value available at that index .

**Example=**list1=[‘physic’,’chemistry’,1997,2000];

list2=[1,2,3,4,5,6,7]:”,list1[0]print”list2[1:5]:”,list2[1:5]

**Output:** list1[0]:physic

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

**Update:** list= [‘physic’,’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

***Inheritance***

* It a allow you to define a new class based on existing class know as the base or parents class.
* New class is called the derived child class.
* Through inheritance ,derived class inherit the attribute & method of the base class & method or overside the inherited
* You can define a class that inherit from another class by mentioning the base class in parenthesis after the derived class.

**Property of inheritance:**

* Multilevel
* Multiple level

***Multilevel:***

It is a types of inheritance that involve inheriting the a class that has already inherited some other class

Class a:

def display a(self):

print(“a class method”)

class b(a)

def display b(self)

class c(b)

def display (“c class method”)

c1=c()

c1.displayc()

c1.displaya()

c1.displayb()

**Output=** a class method

b class method

c class method

***Multiple level:***

It is a types of inheritance that involes inheritance a class that has already inherited some other class

Class one:

def display one(self);

print(“one class method “)

class two:

def display two(self);

print(“two class method”)

class three(one,two):

def display three(self):

print(three class method)

t=three

t.displayone()

t.displaythree()

t.displaytwo()

t2=two()

t2.display two()

**Output=**

One class method

three class method

two class method

one class method

***Polymorphism***

**“One name many form”**

It enable the ability to perform difference action or behaviours based on the actual types of an object at run time.

Methods / functions / operators with the same that can be executed on many object or classes.

**Types of polymorphism:**

Static (compile time) : Overloading

Dynamic(Run time): Overriding

**Dynamic:**

In overriding accure when a derived class define a method with the same name as a method in this super class /base class /parents class.

**Static:**

In python if a method such that it can perform more than one task , its is called method overloading.

**Overriding:**

Class shape :

def are():

pass

class rect(shape):

def\_init\_(self,len,width);

self.len=len

self.width=width

def area (self):

return self .len\*self.width

class circle(shape):

def\_init\_(self r):

self.r=r

def area(self):

return(self.r\*\*2)\*3.14

r=rect(3,4)

print(“area of rectangle:,r.area())

c=circle(2)

print(“area of circle “,c.area())

**Output=**area of rectangle 12

Area of circle 12.56

**Overloading:**

Class mathoperation:

Def area(self ,a=none,b=none)

If a!= none & b!=none:

print(“area of rectangle:”a\*b)

elif a!=none:

print(“area of circle “(a\*\*2)\*3.14

else:

print(“please enter at least one value “)

m=mathoperation

m.area (3,2)

**Output=**

Area of rectangle: 6

***Loops***

A loops is a instruction that repeat multiple time as long as same condition

**There are three types**

For loops

While loops

Nested loop

**For loops:**

For loops is used to iterate a variable over a sequential in order that they appear

**Syntax:**

For<variable>in<sequence>

a=2

for a in range (3,6):

print(“ok”)

b=”razan”

for cin b:

print(c)

tuple=(1,2,3,4,6)

for c in tuple:

print(c)

dict={1:”razan”,2:”noor”,3:”nancy”}

for c in dict:

print©

tuple=(1,2,3,4,6)

for c in tuple :

if c is 3

break

print©

**Nested loop:**

Loops defined within another loops is called nested loops .when an outer loop contain an inner loops in its body it is called nested looping

**#for Nested loop**

for a in range(1,5):

for b in range(1,a+1):

print(a)

**Output=**1

2

2

3

>>>>>.

**While loops:**

While loops is used to execute number of statement or body till the condition passed in while is true .once the condition is false ,the control will come out of the loop

#while loops

a=10

while a>1:

a=a1

Output=10

9

8

7

6

5

4

3

2

>>>>>

***Project***

**Bank Management System:**

import tkinter as tk

class BankAccount:

    def \_\_init\_\_(self, account\_number, account\_holder, initial\_balance=0):

        self.account\_number = account\_number

        self.account\_holder = account\_holder

        self.balance = initial\_balance

    def deposit(self, amount):

        if amount > 0:

            self.balance += amount

            l4.config(text=f"Deposited ${amount}. New balance: ${self.balance}")

        else:

            l4.config(text="Invalid deposit amount. Please enter a positive number.")

    def withdraw(self, amount):

        if amount > 0 and amount <= self.balance:

            self.balance -= amount

            l5.config(text=f"Withdrew ${amount}. New balance: ${self.balance}")

        elif amount <= 0:

            l5.config(text="Invalid withdrawal amount. Please enter a positive number.")

        else:

            l5.config(text="Insufficient funds.")

    def check\_balance(self):

        l5.config(text=f"Account balance for {self.account\_holder}: ${self.balance}")

def register\_account():

    account\_number = t1.get()

    username = t2.get()

    initial\_balance = float(t3.get())

**initial\_balance = float(t3.get())**

**# Create a new bank account for the user and store it in a dictionary with the account number as the key.**

**accounts[account\_number] = BankAccount(account\_number, username, initial\_balance)**

**l4.config(text=f"Account for {username} registered with an initial balance of ${initial\_balance}.")**

**def view\_balance():**

**# Retrieve the account for the current account number and display their balance.**

**account\_number = t1.get()**

**account = accounts.get(account\_number)**

**if account:**

**account.check\_balance()**

**else:**

**l5.config(text="Account not found.")**

**def deposit\_money():**

**account\_number = t1.get()**

**account = accounts.get(account\_number)**

**if account:**

**amount = float(t3.get())  # Use t3 for the deposit amount**

**account.deposit(amount)**

**else:**

**l4.config(text="Account not found.")**

**def withdraw\_money():**

**account\_number = t1.get()**

**account = accounts.get(account\_number)**

**if account:**

**amount = float(t3.get())  # Use t3 for the withdrawal amount**

**account.withdraw(amount)**

**else:**

        l4.config(text="Account not found.")

 l4.config(text="Account not found.")

root = tk.Tk()

root.title("Simple Bank System")

root.geometry('350x200')

accounts = {}  # Dictionary to store user accounts

l1 = tk.Label(root, text="Account Number")

l2 = tk.Label(root, text="User Name")

l3 = tk.Label(root, text="Amount")

t1 = tk.Entry(root)

t2 = tk.Entry(root)

t3 = tk.Entry(root)

b1 = tk.Button(root, text="Registration", command=register\_account)

b2 = tk.Button(root, text="View Balance", command=view\_balance)

b3 = tk.Button(root, text="Deposit", command=deposit\_money)

b4 = tk.Button(root, text="Withdraw", command=withdraw\_money)

l1.grid(row=0, column=0)

l2.grid(row=1, column=0)

l3.grid(row=2, column=0)

t1.grid(row=0, column=1)

t2.grid(row=1, column=1)

t3.grid(row=2, column=1)

b1.grid(row=3, column=1)

b2.grid(row=4, column=1)

b2.grid(row=4, column=1)

b3.grid(row=5, column=1)

b4.grid(row=6, column=1)

l4 = tk.Label(root, text="", fg="green")

l4.grid(row=7, column=1)

l5 = tk.Label(root, text="", fg="blue")

l5.grid(row=8, column=1)

tk.mainloop()

**Output**= Account no: 123456789

User name: varun

Amount: 1000

Initial balance :20000

***CONCLUSION***

The project “PYTHON based Bank Management System” has been successfully designed and tested. Integrating features of a working bank which can perform task like creating account, depositing, withdrawal, dairy entry. It uses the basic of python and libraries, which is understandable easily.