Roll No: 19761 B.Jabnavi

1. Explain Programming and Python in detail

Defination and purpose of programming

Defination: Programming is the process of developing & writing a set of instructions using a Programming language so that a computer can Perform Specific tasks.

Purpose:

- To solve problems using computers
- To perform tasks automatically
- To develop applications. software, website & Systems
- To process data & make decisions
- Improve Speed and accuracy -> computers don't make calculation mistakes like humans
- Process and analyze data -> Ex: weather predictions, business repots. data analysis

Characteristics:

Python has several important characteristics that make it popular and Easy to use

1. Simple and easy to learn: python has clear and easy syntax so that everyone can understand it easily.

2. High level Language: programmers do not need

to manage memory or hardware details
python handles them automatically & even it
allows developers to focus on logic rather
than System-level Operations

3 Interpreted Language: python executes code
line-by-line which makes debugging Easier

4 Object Oriented: Python Supports class and
Object and concepts like inheritance and
Polymorphism

5. platform independent: python programs can run
on windows, linux, & Mac without
modification

6 Dynamically typed: data types are assigned
automatically at runtime, no need to declare
them.

7 Large Standard Library & Ecosystem
Python has a vast collection of pre-written
modules & packages, like math, file handling,
networking etc.

8 Open Source & free: Python can be downloaded
freely & its Source code is open to Everyone.

Applications:

Programming is a versatile skill used across virtually every industry

- Web development : Creating interactive websites & backend systems
- Software Development: Building Operating systems, desktop applications and utilities
- Data Science & Machine Learning (ML): Analyzing large datasets, creating visualizations & developing AI models (eg: using Python libraries like pandas or NumPy).
- Automation: writing scripts to automate everyday tasks like file management or sending emails
- Game development: Building video games etc
- Robotics and Control Systems programming microcontrollers and other hardware for automation in various fields including medical technologies.

Types of Comments in python with Syntax:

Python Supports three types of Comments

1. Single-line Comment

-> used to write a short comment in one-line

-> Starts with the # Symbol

Syntax:

```
# This is a single-line comment.
```

2. Multi-line Comment:

-> used to write long Comments in multiple lines.

-> written using triple quotes ''' ''' & """ """

Syntax

```
'''
This is a
multi-line comment
in Python
'''
```

3. In-line Comment:

-> Comment written at the end of a Statement

-> used to explain a specific part of Code

Syntax

```
x=10 # assigning value 10 to x
Print(x) #printing the value of x
```

Importance of python in modern software development.

Python is very important in today's software world because of it's simplicity, power and wide usage.

Some key points are:

1. Easy to learn:

Python is simple, so developers can write programs faster.

2. Saves times:

Less code is needed, so software is built quickly.

3. used in modern technologies:

Python is widely used in AI, ML, & data science

4. Many Libraries Available:

Python has ready-made tools for web apps, data, AI, games etc.

5. works on all platforms:

Python programs run on windows, linux, & Mac

6. used by big companies:

Companies like Google, Netflix, & NASA use python

7 Large Community:

Many People use python, So helps and Support are Easily available

2) Describe data types and operators in python with Suitable Examples

Built - in data types in Python (Numeric, Sequence, Set, mapping, Boolean)

Data types: A data type tells what kind of value is Stored in a variable. Such as numbers, text true/ false etc.

Python has a different data types

1. Numeric Data types:

Numeric data types are used to Store numbers

There are three types

-> int -> whole numbers (no decimal)

-> float -> Decimal numbers

-> complex -> numbers with imaginary part (a+bj)

Eg

x = 10 #int

y = 3.14 #float

z = 2+3j # complex.

2. Sequence data types:

These Store multiple items in an ordered manner.

Common Sequence types are

a. String (Str):

Stores text. String data types are

in quotes
Eg:
name = "Ronaldo"

b. List: These are ordered, changeable & allows duplicate list are written in "[ ]"
Eg:
fruits = ["apple", "mango", "guava"]

c. Tuple: These are ordered, but unmutable. Tuple uses "( )"
Eg
colors = ("black", "brown", "blue")

3. Set Data Types: Stores data unordered items and no duplicates are allowed. Set data types uses "{ }"
Eg: numbers = {1, 2, 3, 4, 5}

4 Mapping data types:
Stores data in key : value pairs
It uses "{ }". Only dictionary belongs to mapping type
Eg
Student = {"name": "Janu", "age": 21}
here : "name" & "age" are keys

5. **Boolean Data Types**: Stores True & False Values used in Conditions & logic

Eg: Booleans often Comes from Comparision

```
Print (10 > 5) # True
print (2 == 3) # False
```

* **Type identification using type( )**

in Python, you can identify the type of variable or value using the built-in type( ) function It tells you what kind of objects. It is (like int, str, list, etc)

here's a clear Explination along with examples

Syntax: type(object)

-> object -> the variable or value you want check

-> Returns the type of the object

Eg:

1. **Numeric types**:

```
x = 10
y = 4.15
print(type(x))  # <class 'int'>
print(type(y))  # <class 'float'>
```

2. **String** types:

```
name = "Jahnavi"
print (type(name)) # <class 'str'>
```

3. Boolean Type

```
is-valid: True
print (type (is-valid)) # <class' bool's
```

4. List, Tuple, Set, Dictionary

```
my-list = [1,2,3]
my-tuple = (1,2,3)
my-set = {1,2,3}
my-dict = {"a":1, "b":2}
Print(type (my-list)) # <class 'list's
print (type(my-tuple)) # < class' tuple's
print (type (my-set)) # <class 'set)
Print (type (my-dict )) # < class 'dict'>
```

Note: you can also use "isinstance()" if you want to check if a variable belongs to a certain type.

Eg

```
x=10
print (isinstance (x, int)) # True
print (isinstance ( x, float)) # False
```

Various Python Operators

Operator: An Operator is a Symbol that tells the computers to Perform Certain mathematical and logical operations

There are different operators in Python They are.

1. Arithmetic Operators: These are used for doing some basic mathematical calculations

Operator	Meaning	Example	Result
+	addition	10+5	15
-	Subtraction	10 - 5	5
*	Multiplication	10 * 5	50
/	division	10 / 5	2.0
//	floor division	10 // 3	3
%	Modulus	10 % 3	1
* *	Exponent (power)	2 * * 3	8

2. Assignment operators: used to assign Values to Variables

Operator	meaning	example	Equivalent to
=	Assign	x = 5	-
+ =	add assign	x + = 3	x = x + 3

- =	Subtract & assign	x - = 3	x = x - 3
* =	multiply & assign	x * = 3	x = x * 3
/ =	divide & assign	x / = 3	x = x / 3
// =	floor & assign	x // = 3	x = x // 3
% =	mod & assign	x % = 3	x = x % 3
** =	power & assign	x ** = 3	x = x ** 3

3. Comparision (relational operators)

used to Compare values. returns True / False

Operator	Meaning	Example	Result
= =	Equal to	5 = = 5	True
! =	not equal	5 ! = 3	True
>	greater than	5 > 3	True
<	less than	3 < 5	True
> =	greater & equal	5 > = 5	True
< =	less & equal	3 < = 5	True

4. **Logical Operators**: used for logical decisions. works with Conditions.

Operator	Meaning	Example	Result.
and	True if both true	True and false	false
or	True if at least one true	True & false	True
not	reverse the condition	not True	false

5. **Membership operators**: used to test if a value exist in a sequence (list, String, tuple, set)

operator	meaning	example	result
in	True if value present	'a' in apple	True
not in	True if not same object	b is in cherry	True

6. **Identity operators**: used to compare memory locations of two objects.

operator	meaning	Example	result
is	True if Same object	x is y	True/false
is not	True if not Same object	x is not y	True / false

Eg:
```
x = [1,2,3]
y = x
z = [1,2,3]
print(x is y) # True (Same object)
print (x is z) # False (Same value, different
                object)
```

*. Real-world usage of operators:

1. Arithmetic Operators:

used for Calculating things in daily life

Eg: total bill, marks, Salary, distance

2. Assignment Operators:

used to store or update values in a Program

Eg: adding points, updating balance, Counting item.

3. Operatō Comparison: used to compare values

Eg.

-> Checking if age is 18 or not

-> checking if marks are above pass marks

-> checking if stock is available

4. Logical operatōs:

used to combine conditions

Eg.

-> Login only if email AND Password are correct

-> can vote if age >18 AND citizen

5. Membership operatōs.

used to check if two & something in a list

or group exists

-> checking if a name is in contacts

-> checking if an item is in a shopping

cart

6. Identity operatōs:

used to check if two thing are exacly

same object in memory

Eg

Checking if two variables refers to the

game user session

3. Explain python input & output operations in detail

* Input functions & it's default data types Input functions.

* Input functions are used to take input from the user

* Syntax : variable = input ("message")

* It always returns data as string by default

Default Data Types

Data taken using input() is always stored as str (string type)

Type Conversion (casting)

To convert input to other data types.

*) Convert to integer

num = int (input (" enter a number :"))

* Convert to float:

price = float (input (" enter price :"))

* Convert to boolean (manually interpreted)

Eg:

x = input ("enter anything :")

print (type (x)) #output : < class 'str' .

key points:

*) Input function pauses program and waits for

users input
*) useful for interactive programs
*) Requires type conversion for mathematical operations

* Type Conversion while taking input :

*) input always takes data as String (str) by default
*) To use numbers (for math), we must convert the input
*) common Conversions

int() -> converts input to integer
float() -> converts input to decimal number

Eg

```
a = int(input ("enter age :"))
b = float (input ("enter price :"))
```

*) without Conversion

```
x = input ("5") -> "5" (String)
y = input ("3") -> "3" (String)
x+y = "53" (String concatenation)
```

*) with Conversion :

```
x = int (input ("5")) -> 5 (integer)
y = float (input (5.8)) -> 5.8 (decimal)
```

*) input ()->str

int (input()) -> integer

float (input()) -> decimal number

Type conversion is needed for arithmetic operators.

* Taking multiple inputs:

Sometimes we need more than one input from the users

Method-1:

a, b = input(). split()

* user enters values in one line, only.

* default type is string

Eg:

```
a, b = input ("enter two numbers :"). split( )
```

* Type Conversion with split():

if we need integer values:

```
a, b = map (int, input(). split( ))
```

if we need float values :

Eg

```
x, y = map (float, input(). split())
```

Eg.

```
a, b = map (int, input ("enter two numbers :").
        split())

    print (a+b)
```

here, split() -> breaks input by space

map() -> converts each value.

* formatted output using print(), separators, and format specifiers

*) print() is used to display Output.

* 'Sep' changes the Seperator between Values

Eg print ("1, 2, 3, sep = "-") -> 1 - 2 - 3

"end" changes the ending of the line

Eg: print ("Hello", end = " ")

*) format is used to insert Values in a formatted way.

Eg: "Name: { }" format (name)

*) f-strings are used for modern formatting

Eg: f "Name : {name}"

*) format Specifiers:
%d -> integer, %f -> float, %s -> String, %.2f -> 2 decimal places.

4. Discuss Control Statements & Decision making Statements in python

* meaning & importance of control Statements

meaning:

Control Statements are special instructions in a programming language that control the flow of execution of program. They decide which part of the code runs how many times it runs, & under what conditions it runs

Importance of Control Statements

1. Decision making: Control statement allow programs to make decisions & choose different actions based on conditions

2. Repetition of tasks: Loops help execution of a block of code multiple times without writing it repeatdly.

3. Better program flow & control: They allow developers to control the order in which instructions are executed improving efficiency.

4. Reduces code length: using loops and conditions reduce duplicate code, making the program shorter & cleaner

5. Improves flexibility: program becomes more flexible and adaptable to different inputs &

situations
6. Enhances program logic. They make it possible to implement real-life logic in software

Types of Control Statements:
in python control statements are mainly three types

1. Conditions (Decision making statement).
used to check conditions and decide what to do
* if
* if else
* elif

2. Looping (repetition) statements:
used to repeat a block of code multiple times
*) for
*) while

3. Jump statements.
used to change the normal flow inside loops
*) break -> stops the loop
*) continue -> skip one iteration and continues
*) return -> exits a funcion

Decision-making statements : if, if-else, if-elif-else
1. if - statement :
*) used when you want to check one conditions

*) If the condition is true, then the code runs

Eg
```
Oge = 18
 if age > = 18 :
    print ("you are an adult")
```

2.) if - else Statement:

* used when you have two possible outcomes
* If the Conditions is true, One block runs
* If the Condition if false, another block runs

Eg
```
marks = 30
     if marks > = 35 :
           print ("pass")
   else :
           print ("fail")
```

3.) if - elif - else Statement:

*) used when you have multiple Conditions
*) elif means else if
*) if checks conditions one by one
*) only the first true condition runs.

Eg:
```
marks : 75
if   marks > = 90 :
           print ("Grade A")
elif  marks > = 75 :
           print ("Grade B")
```

```
elif  marks >= 60:
        print ("Grade C")
else:
    print ("Grade D")
```

Syntax flow & execution control with examples

1) If Statement:

Syntax:
```
    if condition:
            Statements
```

flow & Execution control:

* The program checks the condition
* if the condition is true the statement inside if are executed
* if the condition is false, the if block is skipped

Eg
```
num=10
 if num>0:
     print(" positive number")
```

2) if else Statement:

Syntax:
```
        if condition:
              Statements
          else:
              Statements
```

flow & execution control:

*) condition is checked
*) if true, if block runs
*) if false, else block runs
*) only one of the block execues

Eg

```
marks = 30
if marks >- 35:
        priue("pass")

else:
    priue ("fail")
```

3) if - elif - else statement:

Syntax

```
    if condition1:
            Statements
      elif condition2:
            Statements
        elif condition3:
            Statements
         else:
            Statements.
```

flow & Execution Control:

* program checks condition from top to bottom
* as soon as one condition is true, its block

runs.

* remaining condition are ignored
* if none are true , the else block executes

Eg

```
marks = 75
if marks >=90:
        print ("Grade A")

elif marks >=75:
        print (" Grade B")

elif marks >=60:
                print ("Grade c")

else:
        print (" Grade D")
```

5. write an easy on python programming fundamentals

* Role of programming in problem Solving

programming plays an important role in Solving problems because it helps us tell the Computer what to do in a clear and Step by Step way

1. Breaking problems into Steps: Big problems can be divided into Smaller & easier Steps. So it becomes Simpler to Solve

2. Doing tasks automatically: programs makes the Computer do work by itself without repeating the Same work manually.

3. Giving fast & correct results: programming Computer follows instructions exactly. So the results are faster and more Accurate than humans

4 Handling big amount of information: programming allows the computer to Store, manage & process large data that human cannot handle, easily.

5. Improving logical thinking: while writing a program, we need to think logically and

plan properly, which improves our problem-solving skills in real life too.

Python Syntax Simplicity & readability

Python is known for having a simple & clean syntax

This means the way we write python code is

-> easy to understand

-> easy to write

-> close to normal english language

1. Simplicity in Syntax: python focuses on writing less code with clear meaning. It avoids complicated symbols & unnecessary rules

for examples, in python we don't need ';' or '{ }' for basic structures

Indentation (Spaces) is used to show blocks of code, which makes it simple

2. Readability: it means the code is easy to read & understand, even after many months. python code looks clean & neat, like plain english instructions.

because of this :-

*) developers can find error easily

*) teams can work together better
*) programs are Easier to maintain

Eg

```
if age >18 :
        print ("Adult")
```

This reads like a Simple English Sentence

*) <u>uses</u> of <u>comments for code documentation</u>:

Comments are notes written inside a program that are not executed by the computers. They are used only for explaining the code to humans

<u>why comments are useful</u>:

1. Explain what the code does: comments helps other to understand the purpose of code.

2. make code Easier to read:- They describe logic in simple words, which improves readability

3) help in debugging: by commenting out certain lines temporarily, we can test or debug the program.

4) Team collaboration: In group projects, comments make it easier for others to read & modify the code

Eg

```
# This function adds two numbers
    def add (a,b):
        return a+b  # returning the sum
```

here

-> the first comment explains the function
-> The Second comment explains the line.

Data types, Operators, and Input /output operations

data types data types define the type of data a Variable can hold. They help the Computer understand how to store and use the data

common data types in python.

-> int / integer : whole number likes 10, -4, 200
-> float : decimal numbers like 3.14, 2.5
-> str (String) : text & characters like "Hi", "Hello"
-> bool (boolean) : True & false.
-> List : collection of item like [1, 2, 3, "hi"]
-> Tuple : similar to list but cannot be changed
-> dict (dictionary) : key - Value pairs like { "name": "Janu", "age": 21 }

different data types help us store different kinds of values of programs.

Operators

Operators are Symbols that perform Operations on Variables & data.

1) Arithmetic Operators.

used for math operations.

+(add), - (sub), * (multi), / (div), // (floor div), % (mod)

** (power)

* Comparision / relational operator: used to compare values

==, !=, <, >, <=, >=

They return True / False.

* Logical Operator: used in decision

and, or, not

* Assignment operator: used to assign values

=, +=, -=, *=, /=

Operators help us perform calculations check conditions and assign values easily.

Input & output operations: Input / output operations allow communication between user & program

* Input:

• input() fun is used to take data from the user.

• whatever we type is taken as string by default.

Eg: name = input ("enter your name:")

*output:

print () -fun is used to display output on the Screen

eg: print ("welcome to python!")

control flow using decision making Statements

Control flow refers to the order in which instruction are executed in a Program. In python decision making statements are used to Control this based on conditions

These statements check a condition and decide which block of code should run

why decision-making is needed

because program often need to

-> make choices

-> Compare Values

-> respond differently based on condition

Decision making Statements in python

*) if Statement

• executes a block only when the condition is True

*) if-else Statement

• executes one block if condition is True

• executes another block if condition is False

Real-world Problems using Python Programming.

1. Movie Ticket Pricing:

```
age = int (input ("Enter your age:"))
is3D = int (input (" Enter 1 if you are watching a 3D movie
        else enter 0:"))
if is3D == 1:
    if age <13:
        print (" Ticket Price is ₹200")
    elif age >13 and age < 59:
        print (" Ticket price with extra ₹50 is ₹300")
    else:
        Print (" Ticket price with extra ₹50 is ₹250)
elif age <13:
    Print (" Ticket price is ₹150")
elif 13 >age <59:
    Print ( "Ticket price is ₹250")
else:
    print (" Ticket price is ₹200")
```

```
Enter your age :25
Enter 1 if you are watching a 3D movie else
        enter 0:1

Ticket price with extra ₹50 is ₹300
```

```
age = int(input("Enter your age:")
is3D = int(input("Enter 1 if you are watching a
    3D movie else enter 0:"))
if is3D = =1:
    if age <13:
        print("Ticket price is ₹200")
    elif age >13 and age <59:
     print("Ticket price with extra ₹50 is ₹300")
    else:
     print("Ticket price with extra ₹50 is ₹250")
   elif age <13:
       print("Ticket price is ₹150")
   elif age>13 and age <59:
      print("ticket price is ₹250")
   else:
       print("Ticket price is ₹200")
```

```
Enter your age 25
Enter 1 if you are watching a 3D movie else enter0:0
Ticket price is ₹250
```

2. College Attendance Rule.

```
Per = int (input ("enter attendance:"))
med = int (input ("enter 1 if u have certificate else 0:"))
if per>=75:
    Print ("allowed for exam..")
elif per>=60 and med==1:
    print ("allowed for exam..")
else:
    print ("not allowed for exam..")
```

```
Enter attendance :55
enter 1 if u have certificate else 0:0
not allowed for exam.
```

3. E-commerce Discount

```
bill=int(input("enter bill amount:"))
isprime =int(input("enter 1 if u are prime member
          else 0:"))
if isprime:=1:
    if bill>=5000:
        bill -=bill*0.25
        print("bill after discount:", bill)
    elif bill>=2000 and bill<=4999:
        bill-=bill*0.15
        print("bill after discount:", bill)
    else:
        print("bill after discount:", bill)
elif bill>=5000:
    bill -=bill *0.2
    print("bill after discount:", bill)
elif bill>=2000 and bill<=4999:
    bill-=bill*0.1
    print("bill after discount:", bill)
else:
    print("bill after discount:", bill)
```

```
enter bill amount:4500
enter 1 if u are prime member else 0:0
bill after discount:4050.0
```

4. Smaxlphone Battery warning

```
Perc = Prnt (input ("entea battory percentage :"))
 is Charging = int (input ("enter 1 if u are Chaaging
             else 0:"))

 if  is charging = = 1:
      Print ("charging ")

 elif   perc >= 80:
       Print ("full")

 elif  perc > 21  and  perc < 80:
     print ("normal")

 else:
    print ("low")
```

```
enter  battery  percentage  : 80
enter  1  if  u are  charging  else  0:0
full
```

5.

Driving licence check

```
age = int(input ("Enter your age :"))
istest = int(input ("enter 1 if passed test else0:"))

 if istest ==1:
        if age >=18:
        print ("eligible")
         else:
            print ("not eligible")
     elif age>= 60 and istest==0:
            print ("eligible")
   else:
        print ("not eligible")
```

```
Enter your age : 68
enter 1 if passed test else 0:0
eligible.
```

6. online food Delivery.

```
order = int(input("enter order amount:"))
isgold = int(input("enter 1 if u are gold member
            else 0:"))
distance = int(input("enter distance :"))
if order >= 500:
    if distance >= 10:
        print(" delivery charged")
    else;
        print ("free delivery ... ")
elif isgold == 1:
    if destance >= 10:
        print ("delivery charged")
else:
    print (" delivery charged")
```

```
Enter Order amount :500
enter 1 if u are gold member else 0:1
enter distance :15
delivery charged.
```

7

Bank Loan Approval

```
Sal = int (input ("enter  salary:"))
CScore = int(input ("enter credit Score :"))
if Sal > 20000:
      if cscore > 700:
      print (" loan approved")

   elif  Sal > 50000:
      print (" loan approved")
   else:
      print (" loan rejected")
   else:
      print (" loan rejected")
```

```
enter  salary: 7850
enter  credit Score: 500
loan  rejected.
```

8. Electricity Bill

```
units = int (input ("enter no of units:"))
 if units < = 100:
     bill = units * 2
  elif units < = 200:
     bill = (100 * 2) + (units - 100) * 3
  else:
    bill = (100 * 2) + (units - 100) * 3 + (units - 200) * 5
   print ("Bill is:", bill
```

```
Enter no of units : 800

Bill is : 5300
```

9. Student Scholarship

```
marks = int (input ("enter marks:"))
Sparent = int (input ("enter 0 or 1:"))
if sparent = = 1 and marks >=85;
    print ("gets scholarship")
else:
    income = int (input ("enter income:"))
    if marks >=85 and income <= 50000:
      print ("get scholarship")
    else:
       print ("No scholarship")
```

```
enter marks: 89
 enter 0 or 1: 0
 enter income :59600
 get scholarship
```

10. Online Exam Result.

```
theory = int(input("enter theory marks:"))
practical = int(input("enter practical marks:"))
total = theory + practical
if theory >= 40 and practical >= 40:
    print("pass")
elif total >= 100:
        print("pass")
else:
    print("fail")
```

```
enter theory marks :86
enter practical marks:24
Pass.
```

11. Hotel Room pricing

```
nod = int (input ("enter no of days:"))
 isweekend = int (input ("enter 1 if weekend
              else 0:"))

 normal = 3000
 weekend = 4000
 bill = 1
     if isweekend = = 1:
           bill = nod * weekend
else:
      bill = nod * normal
if  nod >= 3 :
       dis = bill * 0.15
       fin = bill - dis
        Print (" final bill is :", fin)
```

```
enter no of days: 9
enter 1 if weekend else 0: 0
final bill is : 22950.0.
```

12. Gaming Level unlock.

```
Score = int (input ("enter Score:"))
ispremium = int (input ("enter 1 if premium
            else 0:"))
ischeated = int (input ("enter 1 if cheated else
            0:"))

if score> =100 or ispremium= =1:
    if ischeated = =1:
        print ("access denied")
    else:
        print ("game unlocks next level")
else:
    print ("access denied..")
```

```
enter Score :12
enter 1 if premium else 0:1
enter 1 if cheated else 0:0
game unlocks next level
```

13. Mobile Data usage

```
data = int (input ("enter data used:"))
isunlimited - int (input ("enter 1 for unlimited
        data else 0:"))
is roming: int (input ("enter 1 for roming else 0:"))
if data <= 2 or isunlimited : = 1;
    if isroming = : 1;
        print ("unlimited doesn't work")
   else :
       print ("unlimited data")
 else
     print ("no unlimited data")
```

```
enter data used: 1
enter 1 for unlimited data else 0:0
enter 1 for roming else 0: 1
unlimited doesn't work.
```

14. Office Entry System

```
idvalid = int (input ("enter 1 if id is valid else 0:"))
fingerprint = int (input ("enter 1 for valid fingerprint
        else 0:"))
facescan = int (input ("enter 1 for face else 0:"))
isholiday = int (input ("enter 1 for holiday else 0:"))
if idvalid == 1 and (fingerprint == 1 or facescan == 1):
    if isholiday == 1:
        print ("access denied today is holiday")

    else:
        print ("enter into office")
else:
    Print ("access denied")
```

```
enter 1 if id is valid else 0:1
enter 1 for valid fingerprint else 0:1
enter 1 for face else 0:0
enter 1 for holiday else 0:1
access denied today is holiday
```

15. movie rating display

```
avgrate = float (input ("enter rating:"))
iseditorchoice = int (input("enter 1 if it is editor
        choice else 0:"))
 if  iseditorchoice = = 1:
        print ("recommended..")
    elif   avgrate >= 8.5 :
           Print (" excellent.")
   elif    avgrate >=6.0  and avgrate <= 8.4:
    print ("good")
 else:
      print ("average")
```

```
enter   rating :7.5
enter   1 if   it   is   editor   Choice   else 0:0

    good.
```