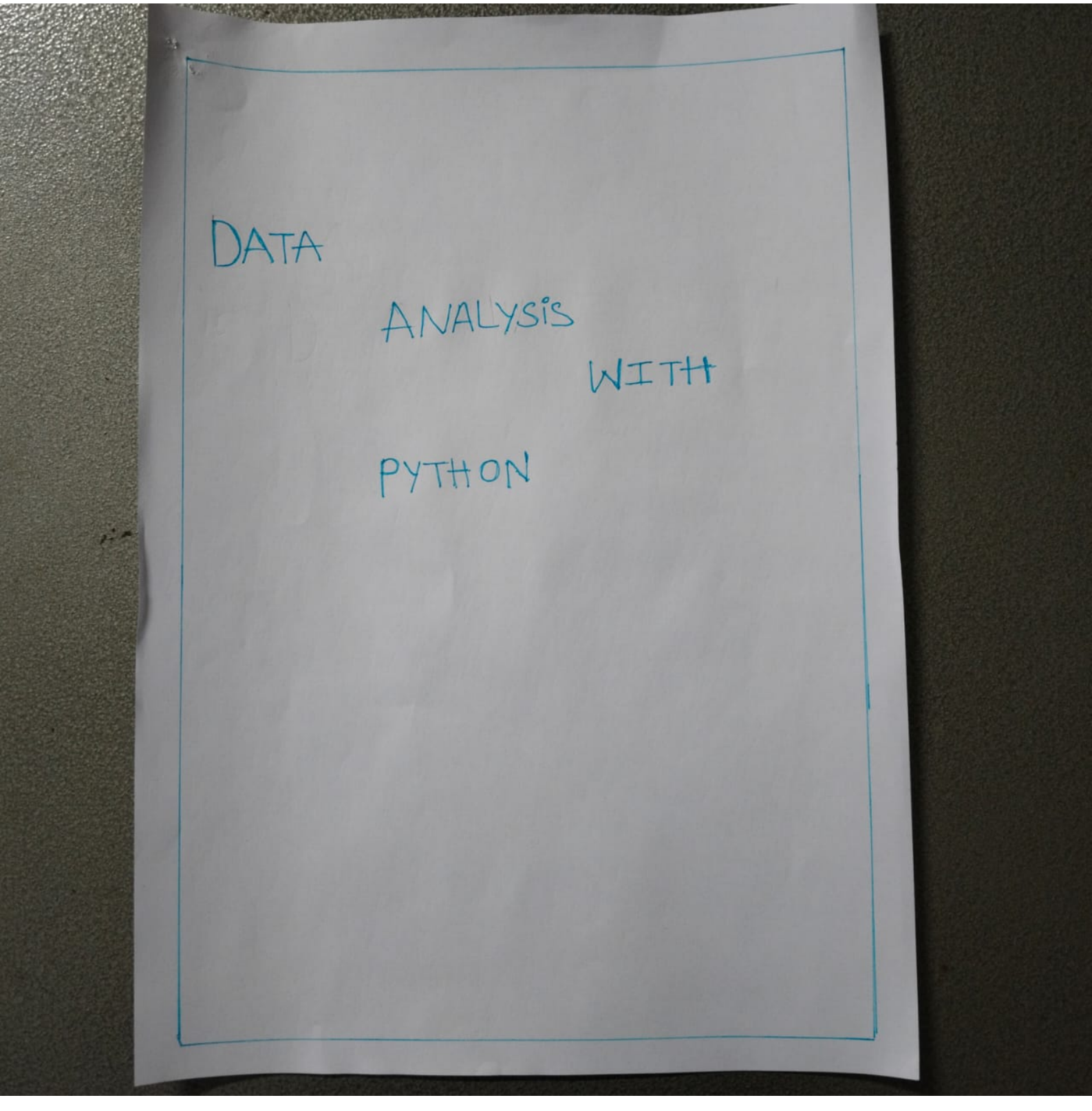


# PYTHON ASSIGMENT

1) Explain programming and python in detail.

**Definition** :- programming is the process of writing instructions that a computer can understand and execute to perform specific tasks, solve problems, or create applications.

**Purpose of programming** :-

The purpose of a programming language is to act as a formal communication system. allowing humans to write instructions (code) that computers can understand and execute to performs tasks, build software. And programming language offers specific syntax and rules for defining designs.

**Python Definition** :-

Python is a high-level, interpreted, general purpose programming language

**characteristics of python** :-

- → Easy to learn and Read
- → Interpreted language
- → Dynamically Typed
- → Object-oriented
- → Extensive standard library
- → cross platform

→ open source & free
→ Scalable & Extensible
→ large community support

**Applications of python :-**

- Data Science & Analytics
- Web Development
- Artificial Intelligence & machine learning
- Automation
- Game Development
- Desktop Application

**Comments in python :-**

→ Comments are non-executable lines used of Explanation
→ Comments are ignored by python & used to explain code.
→ It helps to makes program readable & under-standable.
→ python supports two types of comments

**Single line comments :-** used to comment a single line

**Syntax :-** # This is a single line comment

Multi line comments :-

Importance of python in modern Software Development

Python is a cornerstone of modern Software development due to its simplicity. versatility accross diverse domains, and vast ecosystem of libraries and frameworks. Python's clean and readable syntax allow developers to write and maintain code with ease. This reduces development time, minimizes errors. and improves colloboration among teams. making it deal. for both beginners and experienced professionals.

2. Describe Data types and operators in python with suitable examples.

Built in Data types in python :- python has several built-in data types used to categorize and manage data efficiently.

The Main categories are

- Text Type

→ Str :- Represents a sequence of unicode characters. written inside the quotes

- Boolean Type

→ bool :- Represents truth values, either True or False

- Mapping Type

→ dict :- A dictionary is a built-in data type that Stores values using key-value pairs, making data easy to access and manage

Eg :- Student = {"name": "Sohail", "age": 18, "marks": 90}

- Set Type

→ set :- A set is a built-in data type that Stores only unique values.
written using curly braces (without key-value pairs).

Eg :- numbers = {1, 2, 3, 4}.

Type identification using type() :-

The built-in python type() function is used for identifying the type (class) of an object at runtime

How to use type() for Identification :-

Simply pass the object or variable whose type

You want to known as an argument to the function.

Syntax :- type (object)

object is the value (or) variable whose type want to determine

Example :-
```
x = 10
y = "Hello"
z = [1,2,3]
a = True
print (type(x))
print (type(y))
print (type(z))
print (type(a))
```

Operators in python :- python user special symbols called operators to perform operations on values and variables.

These operators are organized into several categories

- Arithmetic operators :- used for mathematical operators like addition(+), subtraction(-), multiplication(*), division(/), modulo(%), floor-division(//), and exponentiation(**).

Taking user Input :- The input() function is used to get data from the user. It always returns the input as a string data type.

Taking multiple Inputs :-

In python we can take multiple inputs on a single line and convert their types Simultaneously using the map() function or list Comprehension combined with the Split() method.

The map() function applies a given function (eg :- int or float) to each item in an it iterable the list of strings returned by Split()).

Eg :- x, y, z = input ("values :"). Split()

```
print (x)
print (y)
print (z)
```

Formatted output using print(), seperators, and format specifiers :-

To format output using the print() function in python with different seperators and format Specifiers, you can utilize the Sep Parameters

Eg :- x = "Hello World"

- Numeric Type

→ int :- Represents whole numbers without fractions integers and immutable.

Eg :- x = 10
y = -5

→ float :- Represents real numbers with a decimal points

Eg :- x = 3.14

→ complex :- Stores numbers in a+bj format

Eg :- x = 1+2j

- Sequence Type

→ list :- An ordered collection of items, which can be of different data type

Eg :- x = [1, "a", 3.1]

→ Tuple :- An ordered, immutable collection of items. A tuple is a built in datatype used to store multiple values in a single variables.

x = ("Rahul", 18, 75.3)

- Example :- Calculating (price * quantity) + Tax for an online purchase

→ Assignment operators :-

- usage :- Storing and updating data in variables.
- Example :- Tracking a user's score in a game/items in a cart

→ logical operators :-

- usage :- Combining conditions for complex rules in security, game AI, and filtering Data.
- Example :- A Discount applies if (Item is laptop AND credit card is SBI) in an Online Store

3. Explain python Input and output operation in detail.

python input() function :- The input() function in python is used to take input from the user. It waits for the user to type something an key board and once user process enter. it return the value.

- Assignment operators :- Assign values to variables, including combined operations. Such as +=, -=, *=, and |=.
- Comparision operators :- Assign values and return True/false. Example include equal to (==), not equal to (!=), and lessthan or equal to (<=) less than (<), greater than or equal to (>=), and less than or equal to (<=)
- Logical operators :- Combine conditional statements using and, or, and not to return a Boolean Result.
- Identity operators :- check if two operands are the same objects using is and is not
- Membership operators :- Determine if a value is present in a sequence using in and not in.

Real-world usage of operators :-

-> Arithmetic operators :-

- usage :- Calculations in Calculators, e-commerce (total price), finance (interest, profit/loss)

```
    Print ("Unlimited plan doesn't work")
elif data used >= 2 or hasUnlimited plan == 1 :
    Print ("Unlimited data")
else:
    Print ("Limited data")
```

Output :- enter data used : 1.5
Unlimited plan : 1
Roaming : 0
Unlimited data

14. Office Entry System

```
idvalid = int(input (" Is ID Valid :"))
fingerprint = int (input ("fingerprint Matched :"))
facescan = int (input ("Facescan Matched :"))
isHoliday = int(input (" Is it a Holiday :"))
if isHoliday == 1 :
    Print (" Entry is Denied")
elif idvalid == 1and finger print == 1 or face Scan == 1 :
    Print ("Entry Allowed")
else:
    print ("Entry is Denied")
```

```
    print ("pass")
else:
    print ("fail")
```

output :- enter theory :65
enter practical :55
pass

11. Hotel Room pricing

```
is weekend = int (input ("is weekend ? (yes=1,
                                    no=0) :"))
daystayed = int (input ("day stayed :"))
if isweekend == 1:
    Price = 4000
if is weekend == 0:
    Price = 3000
total = price * daystayed
if daystayed >= 3:
    total = total * 15/100
Print ("final bill : ₹", total)
```

output :-
enter is weekend : 1
day stayed :5
final bill : ₹3000.0

```
else:
    print("full")
elif ischar == 1:
    print("charging")
else:
    print("Enter 0 or 1")
```

output:-

```
Enter your battery percentage :87
Enter 1 if phone is charging else 0 : 1
charging
```

5. Driving License check

```
age = int(input("Enter your Age:"))
testpassed = int(input("Enter 1 if you passed the
                test else 0:"))
if (age >= 18 and testpassed == 1) or age >= 60:
    print("Eligible")
else:
    print("Not Eligible")
```

output:-

```
    Enter your Age : 60
```

- mapping type :-

dict - Dictionary, a collection of key-values pairs, enclosed in curly braces

Operators :- Operations are special symbols that performs operations on variables and values.

- Arithmetic operators :- Performs mathematical operations. (+) -> Addition, (-) Subraction, (*) -> multiplication, (/) -> Division, (//) -> Hoor Division, (%) -> modulus (**) -> Exponential
- Assignment operators :- Assign values to variables =, +=, -=, *=, /=
- Comparision operators :- compare two values and return a Boolean result (T/F).

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

- Logical operators :- Combine Boolean expressions, and, or, not
- Identity operators :- check if two variables refers to the same object inmemory

is, is.not.

Output :- Is ID valid : 1

Fingerprint matched : 0

Facescan matched : 0

Is it a Holiday : 0

Entry is Denied

15. Movie Rating Display

```
averageRating = float(input("Enter Rating :"))
isEditorchoice = int(input("Editors choice :"))
if isEditor choice == 1:
    Print ("Recommended")
elif averageRating >= 8.5):
    print (" Excellent")
elif average Rating >=6.5 or average Rating <=8.4:
    Print ("Good")
else:
    print ("Average")
```

Output :-

Enter Rating : 8

Editors choice : 0

Good

Example :-

```
marks = int(input("Enter marks:"))
if marks >=90:
    print("Grade A")
elif marks >=75:
    print("Grade B")
elif marks >=45:
    print("Grade C")
else:
    print("False").
```

5. Write an essay on python programming fundamentals.

Role of programming in problem solving :-

python plays a central role in Problem Solving by providing a language and ecosystem that facilitate the systematic analysis design, and implementation of solutions for a wide range of computational challenges.

- Algorithm Thinking and Design :- python encourages the development of algorithm thinking which involves breaking down a large complex problem into smaller
- Implementation with clear syntax :- python's clean english-like syntax makes the implementation phase a straight forward
- Testing, Debugging and Optimization :- pythons Interpret nature simplifies the Debugging process, as code can be executed line by line and errors caught Early

Output :- enter age : 34
is a 3d movie : 0
final ticket price : 250

2. College Attendance Rule

```
attendance = int (input ("enter attendance:"))
medical certificate = int (input ("Has certificate?
                    (yes = 1, No = 0):"))
if attendance >= 75 or (attendence >= 60 and
                    medical certificate == 1):
Print ("allowed")
else:
Print ("Not allowed")
```

Output :- enter attendance : 75
Has certificate ? (y = 1, n = 0): 1
allowed.

3. E-Commerce Discount

```
bill = float (input ("Enter amount :"))
isprime = int (input ("is prime? (yes = 1, no = 0):"))
if bill >= 5000:
    discount = 0.20
elif bill >= 2000
```

```
discount = 0.10
if isprime == 1:
    discount += bill * 5/100
else:
    discount += 0
final_amount = bill - discount
Print("Final amount:", final_amount)
```

Output :- enter amount : 4000
is prime ? (y=1, n=0) : 1
Final amount : 3799.9

4. Smartphone Battery Warning.

A phone shows :

```
batper = int(input("Enter your battery percentage"))
ischar = int(input("Enter 1 if phone is charging
                    else 0"))
if ischar == 0:
    if batper <= 20:
        print("Low battery")
    elif 21 <= batper <= 80:
        Print("Normal")
```

and f-strings.

The sep parameters in the print() functions allows you to change the separator used between arguments passed to the function. By Default, its a single space ' '.

```
# default seperator (space)
print ("apple", "banana", "cherry")
```

Output :- apple banana cherry

4. Discuss control Statements & Decision-making Statements in python

Meaning of control Statements :-

- control Statements are used to control the flow of execution of a python program
- They decide which Statement is executed. how many times it is executed or when. Execution stops
- using control Statements. we can write decision makings. looping & flow-altering programs.

- Clarifying Logic :- They help to explain complex algorithms or non-obvious logic with in functions.
- Contextual Information :- They can provide context about why a specific approach was choosen over another

Data Types :- python has several built-in data types to stores different kinds of Data

-> Numeric Data types :- used for Numerical values
  - int
  - float
  - complex

-> Text Data types :- used for character sequences
  - str

-> Boolean Data type :- used for logical values
  - bool

-> Sequence Data type :- ordered collection of items
  - list
  - tuple

3. Jump statements.

Decision making statements :- Decision making statements are used to execute code based on conditions. They depend on boolean expression (True/false)

-> If statement :-

- Executes a block of code only if the condition is True
- If the condition is false. the block is skipped

Syntax :- if condition :
    statement.

Example :- age =20
    if age >=18 :
    Print ("Eligible to vote")

-> If-else statement :-

- Executes one block of code if the condition is True and another block (the else block) if it is false.

• Automation :- Python excels at automating repetitive tasks, freeing up time and mental space for more creative and high-level problem-solving.

Problem solving process with python :-

→ understanding the problem
→ Break it down
→ Design and Approach
→ Implement the solution
→ Test & Debug
→ optimize

Use of Comments for code documentation :-

In python, Comments are created using the hash symbol (#) or by using triple-quoted strings. The comments are used to understand the code in a easy way. python offers two primary methods for adding human-readable notes to the code.

Comments are used for explaining the how and why of the code at a granular level.

9. Student scholarship

```
marks = int(input("enter marks:"))
income = float(input("enter income:"))
Single parent = int(input("Is a single parent:"))
if marks > 85:
    Print("Scholarship Allowed")
elif Single parent == 1 or income <= 500000:
    Print("Scholarship Allowed")
else:
    Print("Scholarship Not Allowed")
```

output:- Enter marks :75
enter income : 300000
Is a Single parent : 1
Scholarship Allowed

10. Online Exam Result

```
theory = int(input("enter theory:"))
Practical = int(input("enter practical:"))
if theory >= 40 and practical >=40:
    Print("pass")
elif theory + practical >=100:
```

12. Gaming Level Unlock

```
Score = int(input("enter Score:")
ispremium = int(input("is premium pass:"))
Usercheat = int(input("User cheated:"))
if usercheat == 1:
    Print("access is defined:")
elif Score >= 100 or is premium == 1:
    Print("Level Unlocked")
else:
    Print("Level Locked")
```

Output:- enter score :90
is premium pass : 1
user cheated : 1
access is defined.

13. Mobile Data usage

```
data used = float(input("enter data used:"))
has unlimited plan = int(input("unlimited
                                Plan:"))
isRoaming = int(input("Roaming:"))
if isRoaming == 1:
```

Enter 1 if you passed the test else 0 : 1
Eligible

6. Online Food Delivery

```
amount = int (input ("Enter order amount :"))
isGold = int (input ("Enter 1 if you are a
                Gold member else 0:"))
dis = int (input ("Enter distance in km :"))
if (amount >= 500 and dis <10) or (isGold
                        == 1 and dis <10):
 Print ("Free delivery")
else:
 Print ("delivery is never Free")
```

output :-

Enter order amount = 700
Enter 1 if you are a Gold member else 0:1
Enter distance in km = 11
delivery is never free

7. Bank Loan Approval

```
Salary = int (input ("enter Salary:"))
Credit Score = int (input ("enter Credit Score:"))
```

## Importance of Control Statements :-

- **Decision making :-** They allow a program to evaluate different conditions and choose which actions to take dynamically
- **Code efficiency and organization :-** Loops allows for the execution of the block of code multiple times without redundancy, leading to cleaner, more organized and maintainable code.
- **Flexibility and Adaptability :-** Control statements Enable programs to adapt their behavior in response to changing conditions.
- **Error handling :-** Control structures like try, except, and finally blocks are essential for managing run time errors gracefully. preventing the program from crashing

## Types of Control Statements :-

Python control Statements are broadly classified in three categories.

1. Decision making Statements
2. Loop Statements.

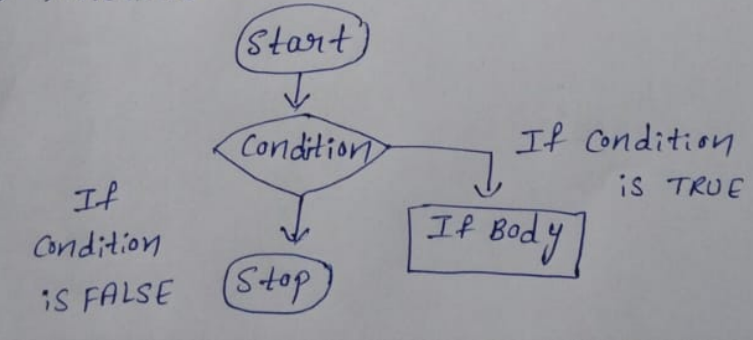
## Input and output operations :-

output :- The print() function is the primary way to display output to the console

Input :- The input() function is used to get user input from the console

## Control flow using decision making statements:

Control flow using decision-making statements directs a program's execution path based on conditions, allowing it to branch or select code blocks to run, similar to real-life choices. key statements like if, if-else, if-else-if and switch evaluate conditions (t/f) to execute specific code segments, while break, continue and return manage flow within loops (or) functions



By default, input() always return data as a String. even if you enter numbers.

Default data types :- Str (string)

Eg :- data = int (input ("Enter a number:"))

* If we need other types like integer (int) or floating-point (float). we need to convert the value explicity.

Type Conversion in python :- Is the process of changing a value from one data type to another. It helps to ensure correct operations and calculations. python supports two types of type conversion.

- Implicit conversion
- Explicit Conversion.

Type Conversion while talking input:-

In python we can create a conversational flow and perform type conversion on user input using the built-in input() function and type-casting functions like int(), float(), Str().

```
if salary >= 50000:
    print("Loan approved")
elif salary >= 30000 and credit score >= 700:
    print("Loan approved")
else:
    print("Loan Rejected")
```

Output :- enter salary : 40000
enter credit score : 400
Loan Rejected

8. Electricity Bill

```
units = int(input("Enter units:"))
if units <= 100:
    bill = units * 2
elif units <= 200:
    bill = (100 * 2) + (units - 100) * 3
else:
    bill (100 * 2) + (100 * 3) + (units - 200) * 5
print("Final bill amount : ₹", bill)
```

Output :- Enter units : 450
Final bill amounts : ₹ 1750

# Real-World problems using python programming.

1. Movie ticket pricing

A movie theatre charges

₹ 150 for children (age < 13), ₹ 250 for adults (age 13-59)

₹ 800 for seniors (age > 60)

If the person is watching a 3D movie, add ₹50 extra.

```
age = int(input("Enter age:"))
_3d = int(input("its a 3d movie:"))
if age <= 13:
    Price = 150
elif age <= 59 and age >= 1:
    Price = 250
elif age >= 60:
    Price = 200
if _3d == 1:
    Price += 50
else:
    Price += 0
Print("final ticket Price:", Price)
```

Syntax :- if Condition :
    #code executed if Condition is true
else :
    #code executed if Condition is false

Example :- num = int (input ("Enter a number :"))
if num % 2 == 0 :
    print ("Even number")
else :
    print ("odd number")

—> if - elif - else statement :-

• used to check multiple conditions sequentially the elif allows you to test for additional conditions without excessive indentation

Syntax :- if condition 1 :
    # code executed if Condition 1 is true
elif Condition 2 :
    # code executed if Condition 2 is true
elif Condition 3 :
    # code executed if Condition 3 is true
else :
    # code executed if all above conditions are false.