Software Programming

Software Programming is the process of writing a set of instructions for a computer to perform a specific task. These instructions are written in a programming language that the system can understand and execute. Programming language are generally classified as low level, mid level and high level.

- Low Level (Binary) : written in 0s and 1s, closest to machine language, very hard for humans to understand.
- Mid Level (Assembly) : Uses mnemonics and symbols, easier than binary but still hardware dependent.
- High Level : Human-readable, portable and easy to write and debug (Eg :- Python, C, Java).

Python

Python is a high-level, interpreted, object-oriented language that uses indentation for code readability. It is dynamically typed, meaning datatypes are assigned automatically at runtime.

Python was developed by Guido van Rossum, and has grown into one of the most popular languages.

- Feb 1991 : Python 1 was launched with basic features.
- Oct 2000 : Python 2 was released with many improvements.
- Dec 2008 : Python 3 was introduced with modern features and better support.

Features of Python

- Easy to learn
- Open Source
- Platform Independent
- Extensive Library
- Indented Programming
- Strong Community Support

Applications of Python

- Web applications
- Mobile apps
- Desktop Applications
- GUI (Graphical user Interface)
- Gaming Applications
- Deep learning
- AI, Robotics, IoT
- Graphical Design

Advantages of Python

- Simple and Readable
- Versatile
- Rich Libraries
- Cross Platform
- Strong Community

Disadvantages of Python

- Slower Speed
- High Memory Usage
- Mobile Development Limitations
- Runtime Errors
- Less Suitable for low level Programming

Comments

comments are part of code that help explain logic but are not executed by the compiler / interpreter. They make programs easier to understand and are useful for documentation.

Types of comments in Python:

1) Single Line Comment : Starts with # and applies only to that line.

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

2) Multi line Comment : Written inside ''' ... ''' or """ ... """.

''' This is a
multi line comment ''',

Keywords

Keywords are reserved words in Python that have predefined meanings and purposes. They cannot be used as identifiers (variable names, function names, etc).

Eg:- True, False, For, while, if, else, def, return, class, try, except.

Variables

A variable is a name or place in memory where a value is stored. It acts as a reference to data and allows easy manipulation during execution. In Python, variables are dynamically typed (no need to declare type explicitly).

Valid Variable

a=7

A=2

name = "Priyanka"

Stuname = "Alicia

Stu_name = "Karan"

num 1 = 9

NUM_2 =15

- stu_id =456

Invalid Variables

emp name = "kamala" # space not allowed

1 num = 123 # cannot start with numbers

emp@id $ = "emp12" # special characters not allowed

Initializing multiple variables in single line.

```
a=4; b=8; c=1;
print (a,b,c)        # output : 4 8 1
```

- Python allows assigning values to multiple variables in single statement. This helps in writing cleaner code, and reduces the number of lines.

Data Types in Python

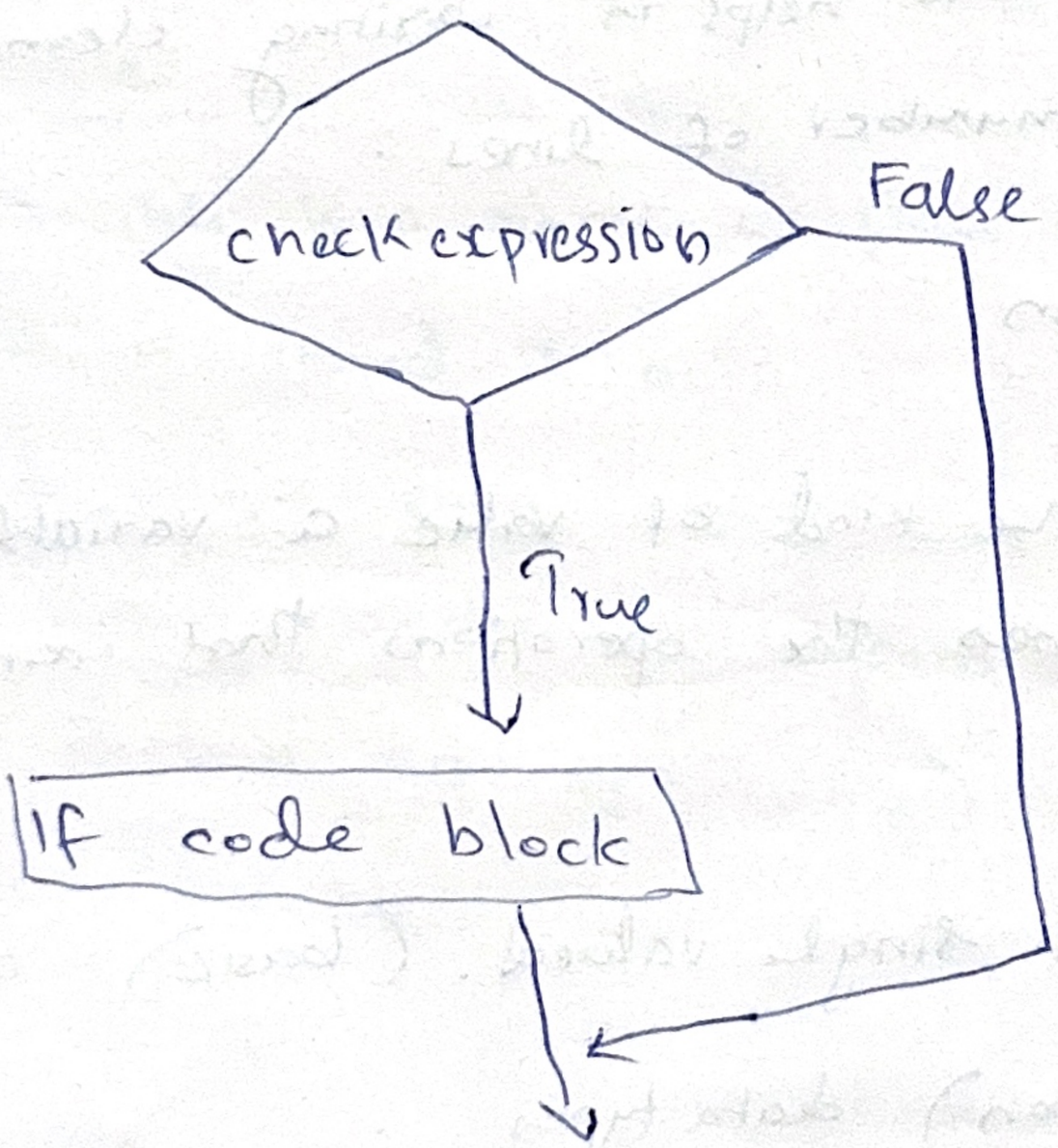
A datatype defines the kind of value a variable can hold and determines the operations that can be performed on it.

Python supports both single valued (basic) and multi valued (collection) data types.

Single Valued Data Type :- int
Float
complex
str

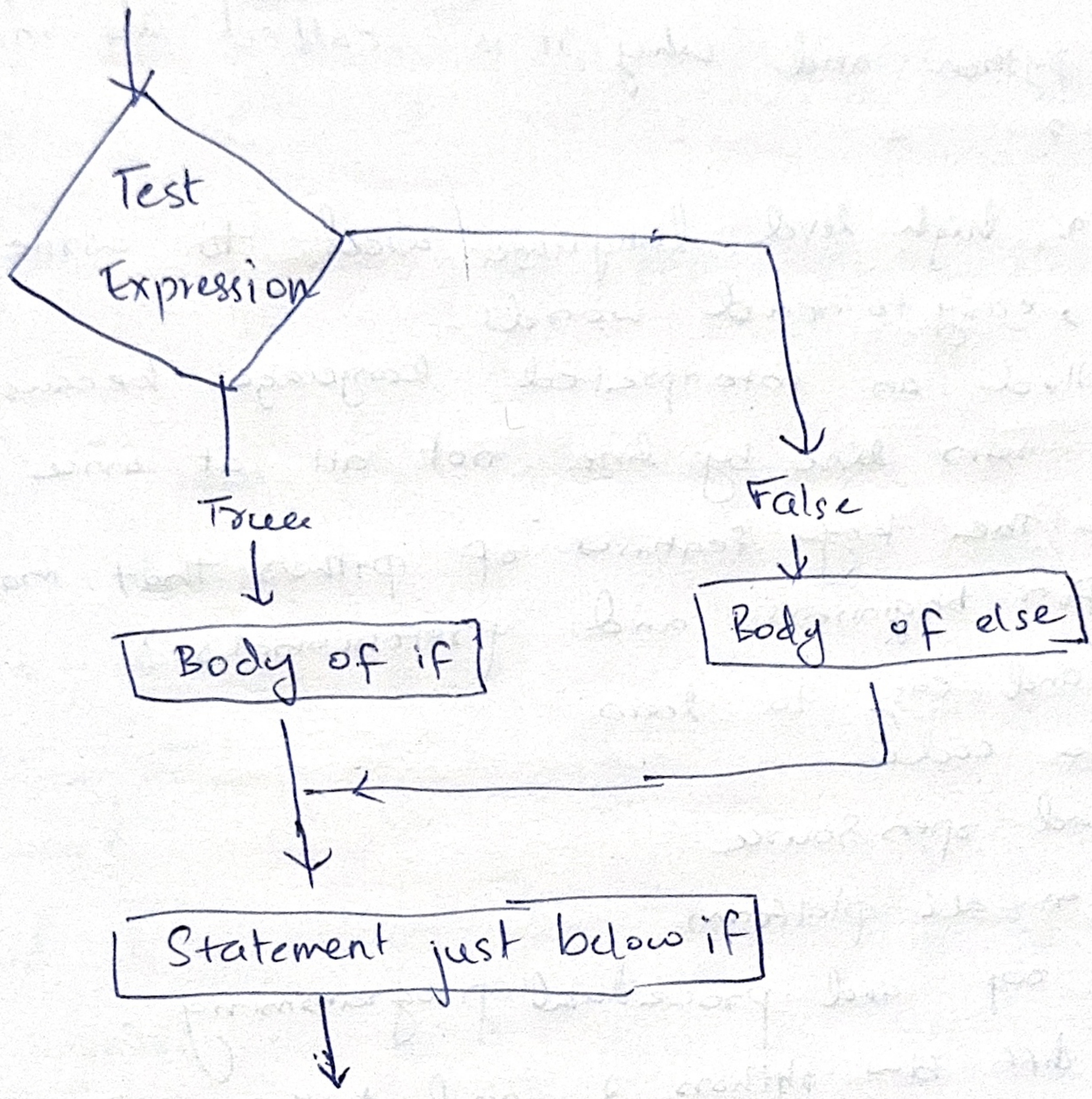
Multivalued Data Type :- list
tuple
set
dict
str

If conditional Statement flow control

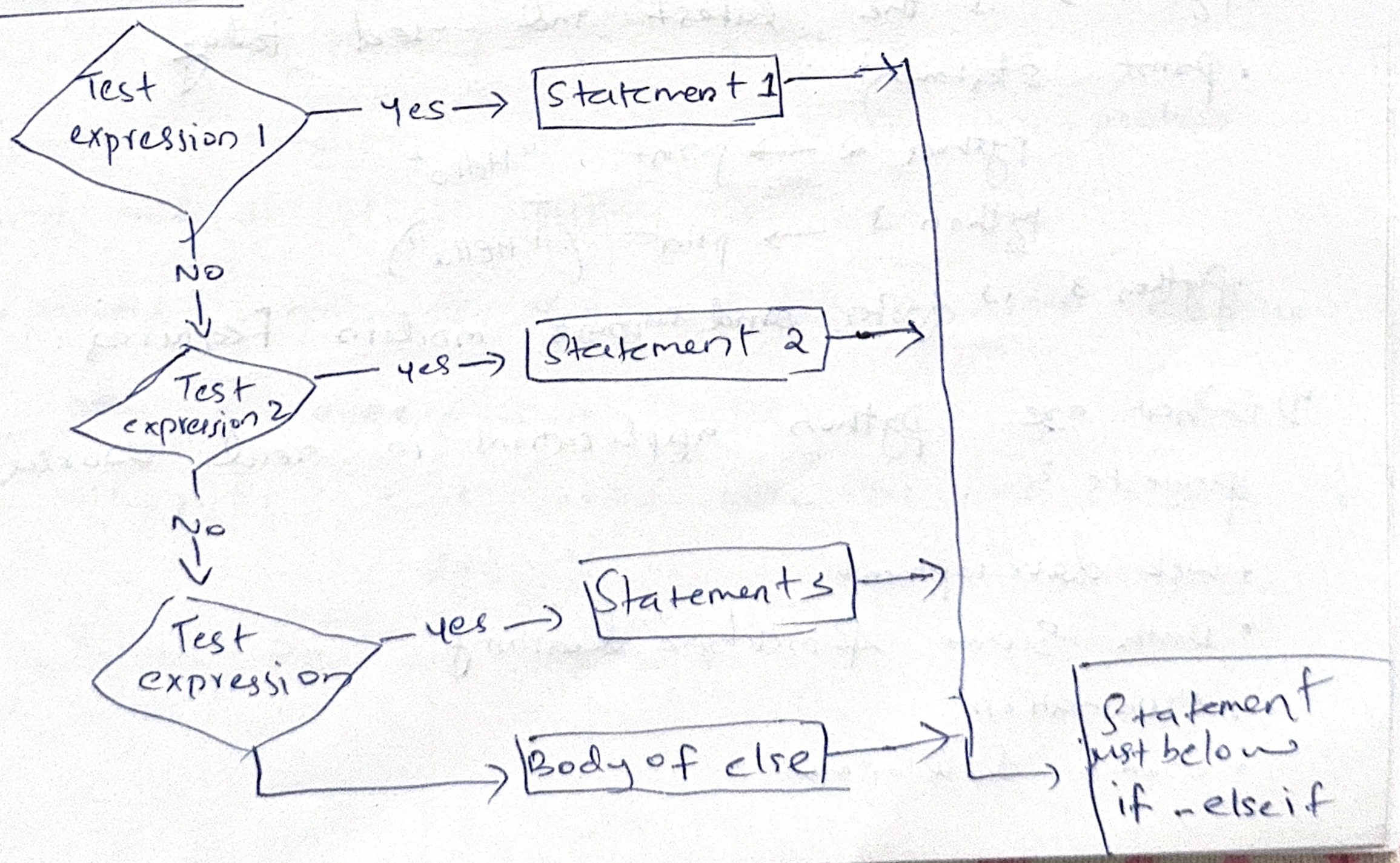


If it is true that goes to statement when false outside the loop

if-else flowchart



Flowchart of elif Statement



Input Functions

- The input function is used to take user input from keyboard.
- It always returns input as a string, even if you type a number.

Eg:
```
name = input("Enter name:")
print("Hellow", name)
```

Print () Function

- It is used to display output on screen. ~~Syntax~~

Syntax:

⇒ Simple print
```
print("Welcome")
```

⇒ Printing variable
```
print("value of x is", x)
```

⇒ f-string (modern & recommended way)
```
name = "Belicia"
print(f"Hello, {name}")
```

⇒ using format ()
```
marks = 85
print("Score obtained {} marks".format(marks))
```

conditional Statement

Execute code only if the condition is true

if Syntax

```
if condition :
    statement
```

eg:
```
age = int (input ("Enter age :"))
    if (age >=18)
        print ("Eligible for vote")
```

if else syntax

```
if condition :
    true statement

else :
    false statement
```

eg:-
```
num = int (input ("Enter a number :"))
    if (num % 2 ==0) :
        print ("Even no")

    else : print ("odd no")
```

Elif syntax

```
if condition
    statement cond 1

elif cond 2 :
    statement cond 2

elif cond 3 :
    statement cond 3

else :
    statement (if all cond are False)
```

eg:- marks = int (input ("Enter marks:"))

```
if (mark >=90) :
    print ("grade A")
elif (mark >=80) :
    print ("grade B")
elif (mark >=60) :
    print ("grade C")
else: print ("grade fail")
```

Nested if else syntax

```
if cond 1:
    if cond 2:
        #code if both cond true
    else:
        #code of cond 1 ~~true~~ but cond 2 false
else:
    #code of cond 1 false
```

eg:- num = int (input ("Enter a no :"))

```
if (num >=0) :
    if (num ==0) :
        print ("No is zero")
    else: print ("No is true")
else: print ("No is -ve")
```