Python — Guido Van Rossum, 1991

- Python is a high level, interpreted, object oriented, dynamically type programming language that uses indentation for code structure.

* High level programming - Easy to understand, closer to human language
* Interpreted - Code runs line by line, no need for compilation.
* Object - oriented - Organise code using objects & classes.
* Dynamically typed - No need to declare variable types. It figures out while running.
* Indented - Giving spaces or tabs at the beginning of a line code.

Features

* Simple syntax - Easy to read & write.
* Open source - Free to use, modify & distribute.
* Libraries - Already have ready-made tools
* less memory - Python manages memory automatically, reducing manual memory management.
* Platform independent - Python code runs on multiple operating system.

Application

- Web development
- Gaming
- ML & AI
- Data Analytics
- Cyber Security
- Deep learning
- Automation

Comments :

- Comments are parts of code that aren't executed.
- Comments can be used to explain python code.

They are used to :

- Explain code - Helps others understand what the code does
- Provide context - Add notes or reminders

Types

1. Single - line comment

   Start with ##

Eg :- ## my, first program

   ## This is a single - line comment.

2. Multi - line comment

   Use triple quotes """ """ or ''' '''

Eg :- ''' write a program to calculate student result '''

   """ write a program to calculate student result """

Keywords

- They are special reserved word that have specific meaning & cannot be used as identifiers, variable names, function name etc

Eg :- True, False, try, for, else, if, while, and, or, not, etc.

Data Types

- A data type defines the kind of value a variable can hold.
- It tells python how the data should be stored & what operations can be done on it.

Common data types in Python.

1. Numeric Types
   - Int -> integers (whole no)
   - float -> decimal no.
   - complex -> Complex no : (3+4j)

2. Boolean Type :
- bool (true or false)

3. Sequence Types :
- str (string)
- list (ordered collection)
- tuple (ordered, immutable collection)

4. Mapping Type :
- dict (dictionary, key - value pairs)

5. Set Types :
- set (unordered collection of unique elements.

Variables :

- A variable is a name used to store & work with data.

Rules to declare variable :

* Valid variable declaration : Eg :-

a = 2
A = 4
num = 20
num1 = 31
num2 = 90
num 3 = 12

StuId = 123
Stu-Id = 127
~~Stu-Id = 1~~
- Stu-name = "Inchara"
• Multiple values in single line
* a, b, c = 4, 6, 1

NOTE
• variable must start with a letter (a-z, A-Z) or an underscore (_).
• It cannot start with a number
• It can contain letters, digits, & underscore.
• No space are allowed
• It should not be a key (like if, for, while, class etc)

* Invalid variable declaration :

Eg :-
Iname = "~~Ra~~ Inchara" (cannot star with a number)
Student name = "Anu" [space not allowed)
class = 10 ('class' is a keyword)
my - age = 25 ('-' is not ~~va~~ allowed)
@value = 100 (special characters not allowed)

Eg :- Data type - Collect ~~to~~ a persons data

- name - str
- age - int
- phn no. - str
- email id - str
- address - str
- insta id - str
- height - float
- weight - float
- education - str
- work exp - ~~str~~ float
- salary - float
- Company name - str
- DOB - str
- gender - str
- blood group - str

Function

1. Input function (Input()) ~~2. Output fun.~~
   - gets data from user.
2. Output function (print()).
   - Shows data to user.

```
Print ("hello good morning")
name = str (input ("enter your name :"))
age = float (input ("enter your age :"))
Print (name, type (name))
print (age, type (age))
```

Methods of Output formatting in Python

1. Comma method / Concatenation (Basic print())

- Can separate text & variables using commas (,) inside print()

Eg 2: format method

- Use {} as placeholders & then apply .format()

3. f-string method

- Put f before the string & use variables inside {}.

Eg in:- form of a question employeeid, employee name, employee salary.

1. print ("employeeid =", employeeid, "\nemployeename =", employeename, "\n employee salary = ", employeesalary, "Rs")

2. print ("employeeid = {} \n employeename = {} \n employeesalary = {} Rs.\n". format (id, name, salary))

3. print (f"employeeid = {employeeid} \n employeename = {employeename} \n employeesalary {employeesalary} Rs.")

Operators

- Operators are used to perform operations on variable & values.

Eg; a = 10
b = 5
Print (a+b)

where, + is an Operator
a & b is a Operands

Types of Operators

1. Arethimetic Operators - are used with numeric values to perform common mathematical operation.

Operators	Name	Eg
+	addition	x + y
-	substration	x - y
*	multiplication	x * y
/	Division	x / y
%	modulus	x % y
**	Exponentiation	x ** y
//	floor division	x // y

2. Comparision Operators - are used to compare 2 values

Operators	Name	Eg
==	equal	x == y
!=	not equal	x != y
>	greater	x > y
<	less than	x < y
>=	greater than or equal to	x >= y
<=	less than or equal to	x <= y

3. Logical Operators - are used to combine conditional statements.

Operators	Description	Eg
and	Returns True if both statements are True	x < 5 & x < 10
or	Returns True if one ~~both~~ ~~are~~ statements is true	x < 5 or x < 4
not	Reverse the result, returns if false if the result is true	not (x < 5 & x < 10)
xor	all conditions should be same	
xnor	all conditions should be different.	

4. Assignment Operators - are used to ~~as~~ assign values to variable.

Operators	Eg
=	x = 5
+=	x += 3
-=	x -= 3
*=	x *= 3
/=	x /= 3
%=	x %= 3
//=	x //= 3

5. **Identity Operators** - are used to compare the objects, not if they are equal, but if they are actually the same object, with same memory location.

Operators	Description	Eg
is	returns true if both variables are the same object	x is y
is not	returns true if both variables are not the same object	x is not y

6. **Membership Operators** - are used to test if a sequence is present in a object.

Operators	Description	Eg
in	Returns True if a sequence with the specified value is present in the project	x in y
not in	Returns True if a sequence with the specified value is not present in the project	x not in y

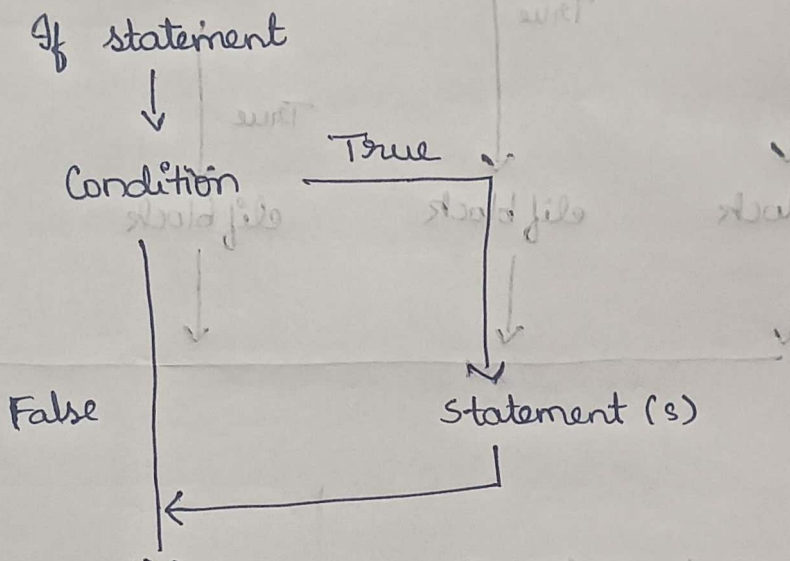
Conditional Statements :-

- It is used make decision in a program.
- Allows us to make decision in code.
- They check conditions expression that result is True or False and execute different blocks of code accordingly.

Types

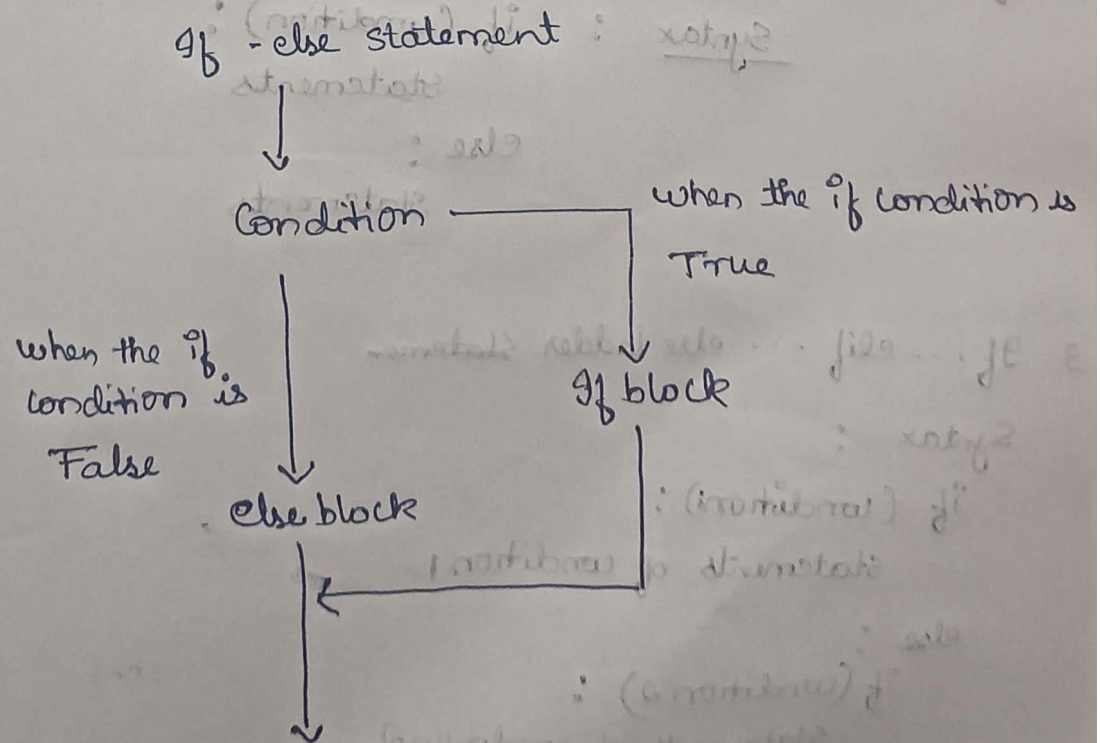
1. If statement

- execites a block only if the condition is True.



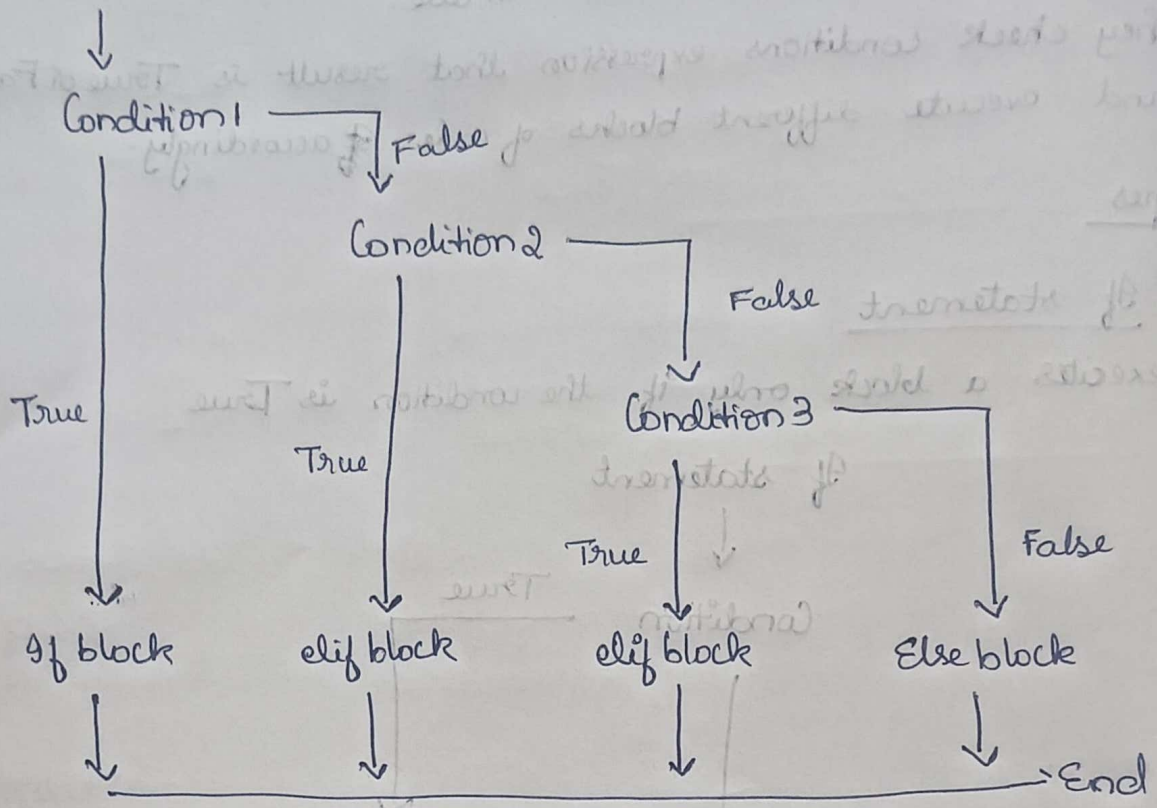
2. If .. else statement

- provides 2 paths : one if condition is True another if False



3. If . . elif . . else ladder

- multiple conditions checked one by one.



1. If statement syntax :

```
if (condition) :
    statements
```

2. If .. else statement

Syntax :

```
if (condition) :
    statements
else :
    statements
```

3. If . . elif . . . else ladder

Syntax :

```
if (condition1) :
    statements of condition 1
else :
    if (condition 2) :
        statements of condition 2
```

```
else :
    if (condition 3) :
        statement of condition 3

    else :
        else block statements.
```

Now used :-

```
if (condition 1) :
    statements of condition 1
elif (condition 2) :
    statement of condition 2
elif (condition 3) :
    statement of condition 3
else :
    else block statement
```

4. Nested if - using one if inside another.

Eg :-

1. Write a program to check the given number is even or odd.

```
n = int(input("enter your value:"))
if (n %.2 = 0) :
    print ("even")
else :
    print ("odd")
```

2. Write a program to check the given input is positive, negative or zero.

```
n = float (input ("enter the numbers"))
if (n>0) :
    print ("positive")
elif (n.<0) :
    print ("negative")
else :
    print ("zero")
```