Python (.py) → extension

↳ Python is an interpreted, object-oriented, high-level Programming Language with dynamic semantics.
↓ runtime pe check

↳ Guido Van Rossum is the founder of Python Programming

↳ Python Syntax are easy as compared to other Languages.

History of Python

↳ Python was introduced by Guido van Rossum in 1989.

↳ First version of Python released in 1991.

↳ In 1994, Python 1.0 was released with new feature which map, filter, Lambda function.

↳ Python 2.x (2000) also add some more feature like Comprehensions, garbage collection System.

↳ After Python 2.x Python 3.x released in 2008.

Features of Python.

↳ Easy to learn and implement.

↳ Open-source. (free)

↳ Broad Standard Library.

↳ cross Platform. (किसी भी OS में काम कर सकते हैं)

↳ Work on interpreter logic.

↳ Multi-Paradigm Language. (Structual or OOPS)

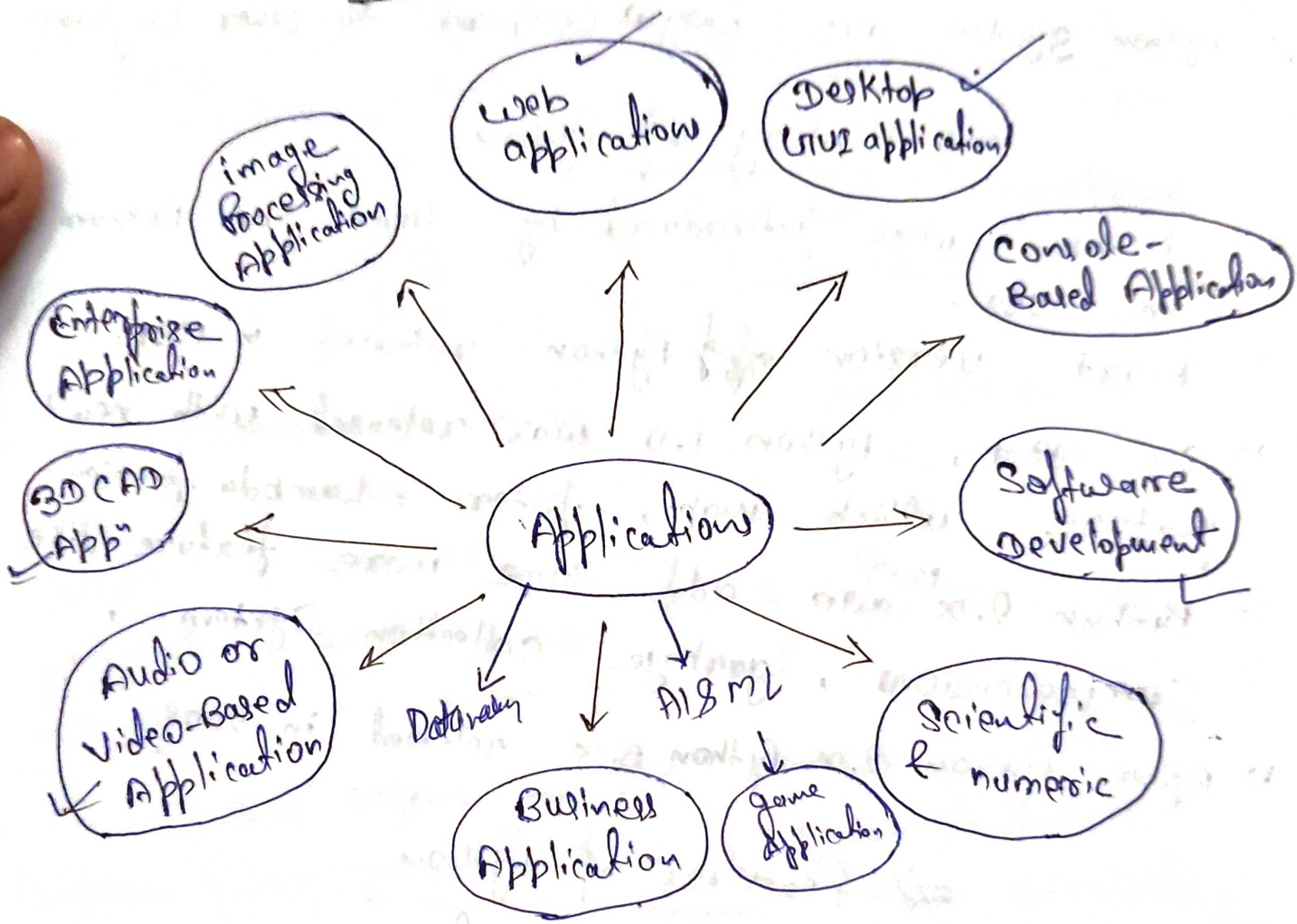
↳ High-level Programming.

↳ Extendable Language. (Combined with another language) & run the code / program.

↳ Expressive Programming Language. (Program & code को express कर सकते है। कम लाइन में कोड लिख सकते है।

↳ GUI Programming Support.

Application of Python.



1) Web Applications :- We can use Python to develop web applications. it Provides libraries to handle internet protocols such as HTML and XML, JSON, Email Processing, request, beautifulsoap, Feedparser etc. One of Python web-framework name Django is used on instagram.

2) Desktop GUI Applications :- The GUI stands for the graphical user interface. Python provides a TK GUI library to develop a user interface. Some popular GUI libraries are given below :-

- Tkinter or TK.
- Kivy (used for writing multitouch applications).

3) Software Development :- Python is useful for the software development process. It works as a supporting language and can be used to build control and management, testing etc.

4) Business Applications :- Business Applications differ from standard applications. E-commerce and ERP are an example of a business application.
↳ Odoo is an example of the all-in-one Python-based application.

5) Audio or Video-based Applications :- Python is flexible to perform multiple tasks and can be used to create multimedia applications. Some multimedia applications which are made by using Python are TimPlayer, cplay etc.

6) 3D CAD applications :- The CAD (computer-aided design) is used to design engineering related architecture. It is used to develop the 3D representation of a part of a system. Python can create a 3D CAD application by using the following functionalities :

- Fandango (Popular).
- RCAM.

7) Game development :- video games are one of the most popular forms of entertainment in the world.

Python Language Used

The list of companies that use Python include the following.

1) Google.
2) Yahoo! (for Yahoo! Maps)
3) Linux weekly News.
4) youtube.
5) Blender.
6) Dropbox.
7) Netflix
8) Instagram
9) Quora

Q. Limitations of Python

1) Speed :- slow
2) High memory Consumption
3) Skills to learn the inbuilt library of Python.
4) Runtime error.

Some Basic Points :-

extension of Python file .py.

Pycharm
IDLE → integrated developing learning environment
Notepad (file)

Python 2nd function
↳ Method etc

Flavours of Python :-

1. Jython
2. PyPy
3. Anaconda Python
4. CPython (core Python)

Some Basic Points & Small Programs of Python.

Python	C
input =>	scanf
Print =>	Printf

* << => Insertion Operator
  >> => Extraction Operator

* All the lines should be in Parallel to 2nd & so on --- lines.

  eg:- { a =
  b
  Print } ← Same/equal ✓

  { a =
  b =
  Print } ← wrong ✗

* : => colon
* _ => underscore
* * => Asterisk
* & => Ampersand
* % => Modulus
* ** => Exponential (Performs Power Calculation on operators)

* * => We in 'list' in the Python for Repeating the elements.

* ~ => Complement

~~* Var = eval (input ("enter data"))~~
~~Print → (Default type of data~~

Video Note:-

```
var = eval(input("enter data: "))   // eval is use for all data type (int, float, ~~bool~~, ~~string~~)
print(var)
print(type(var))
```

Output

~~var = Mehul~~
enter data: Mehul
Mehul
'string'

Output

enter data: 12.89
12.89
'float'

P.T.O //

↳ eval function लिखने के बाद हमे बार-बार int, float etc लिखने की need नहीं पड़ती है।

Python Comments.

↳ Comments are nothing but ignorable part of Python Program that are ignored by the interpreter.

↳ It enhance the readability of code.

↳ There are 2 types of Comment:-

i) Single line Comments:-

↳ Python single-line comment starts with the hashtag symbol (#). If the comment exceeds one line then put a hashtag on the next line and continue the comment.

ii) Multi line Comments:-

↳ A multiline ~~String~~ in Python begines with either three single quotes or three double quotes.

↳ Python does not really have a syntax for multiline comments.

↳ It ~~starts~~ symbol → """ ---- ---- ---- """ or ''' ---- ---- ---- '''

keyword in Python.

↳ Keyword are the reserved words whose meaning already defined in the Python interpreter.

Note:- 1) we can't use a keyword as a variable-name, as a method-name or any other identifier.

2) Python Keywords are the Case sensitive.

(Any version) ← 3) Total 33 keywords.

↳ on Python 3.10.5 version, there are 33 keywords.

↳ Those keywords are:—

['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'Pass', 'raise', 'return', 'try', 'while', 'with', 'yield'].

#variables

↳ Variable is the name of memory location where we can stored different types of values.

eg:- a = 10;
a = 'Mehndi'; # (String value)
↳ data type

variable ← Print (a)

↳ A variable can hold diff^n types of values.

↳ a1 = 20 ✓
Print (a1)

1a = 20 ✗ → error
Print (1a)

↳ a_b = 35
Print ('a_b')

Output
a_b (quotation के अंदर लिखा है वही Print हो जाएगा)

Data types in Python.

Variables can hold values and every value has a data-type. Python is a dynamically typed language hence we do not need to define the type of the variable while declaring it. It does implicity with its type.

eg:- a = 5 ✓

int a = 5 ✗

↳ The variable 'a' holds integer value five and we did not define its type. Python interpreter will automatically interpret variables 'a' as an integer type.

* type() function

↳ Python provides us the type() function which returns the type of the variable passed

↳ It returns the data type of the given values.

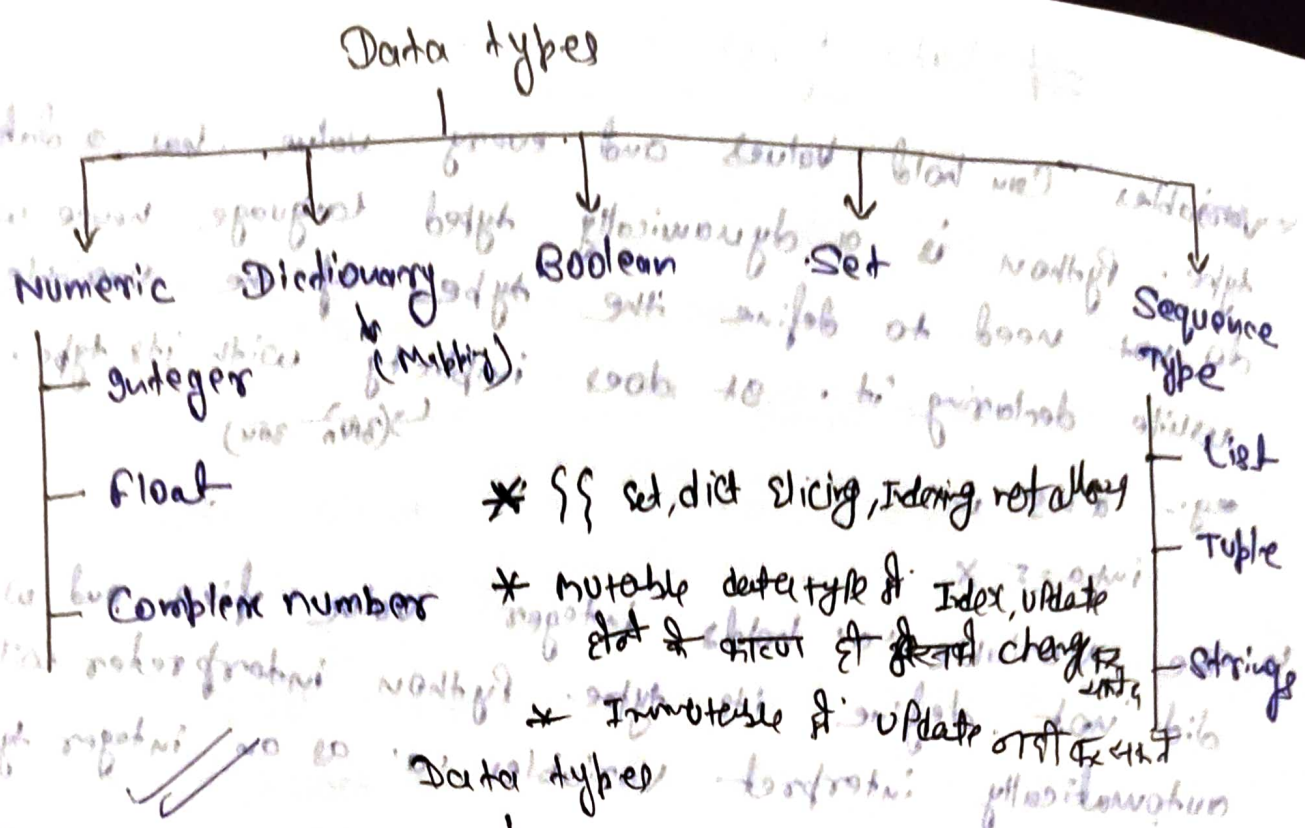
↳ Consider the following example to define the values of different data types and checking its type.

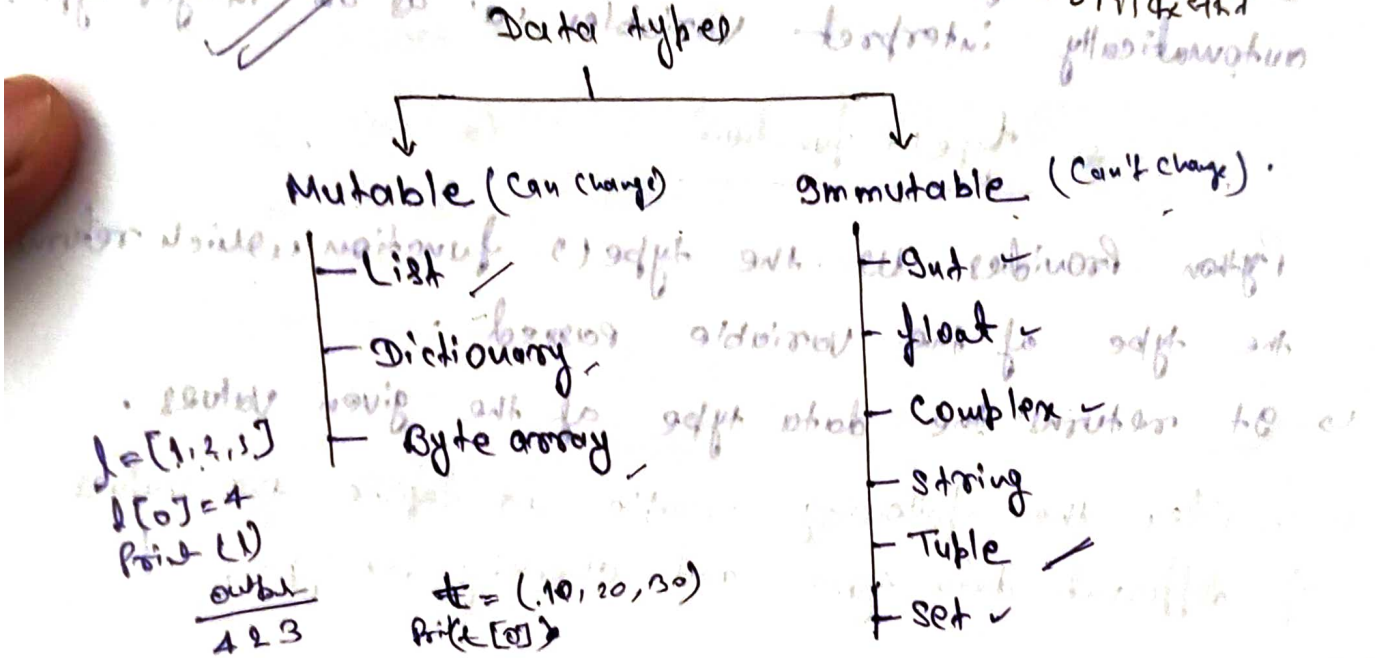
eg:-
```
a = 5
Print (a, type (a))
b = 'Mehudi'
Print (b, type (b))
c = 10.5
Print (c, type (c))
d = 2 + 5j
Print (d, type (d))
e = True
Print (e, type (e))
```

Output

```
5 int
Mehudi str
10.5 float
2+5j complex
True bool
```

Type Conversion & typecasting
↓ (implicit) ↓ (explicit).





* Mutable data type

↳ Mutable object can change its state or contents. It is changable.

↳ In mutable, anything can be modified or changed.

* Immutable data type.

↳ Immutable object cannot change its state or contents. It is not changable.

↳ Immutable is a state which no change can occurs over time. In this, we can't change its value every time.

* Numeric

↳ Number stores numeric values. The integer, float, and complex values belong to a Python Numbers data-type.

↳ Python provides the types() function to know the data type of the variable.

* Python supports three types of numeric data.

1) int :- integer value can be any length such as integers 10, 2, -20, -150 etc. Python has no restriction on the length of an integer. its value belongs to int.

2) float :- float is used to store floating-point numbers like 1.9, 9.902, 15.2 etc. it is accurate upto 15 decimal points.

3) Complex :- A complex number contains an ordered pair, i.e $x + iy$ where x and y denote the real and imaginary parts, respectectively.

↳ It can be 2.14j, 2.0 + 2.3j etc.

Example:-

```
a = 5
Print (a, "is a type of ", type (a))
b = 5.56
Print (b, "is a type of", type (b))
c = 2+5j
Print (c, "is a type of", type (c))
```

Output

```
5 is a type of int
5.56 is a type of float
2+5j is a type of complex
```

* Sequence data type.

1) String :- (immutable)

↳ The string can be defined as the sequence of characters represented in the quotation marks.

↳ In python we can use single, double or triple quotes to define a string.

↳ Multi line strings can be denoted using triple quotes ''' or """ .

eg:-
```
S = 'Hello@123'
Print (S, type (S))
```
Output
Hello@123 <class 'str'>

eg:-
```
S = '''
    Hello
    How are you?
    '''
Print (S, type (S))
```
output
Hello
How are you? <class 'str'>

eg:-
```
S = "10"
Print (S; type (S))
```
output
10 <class 'str'>

* String Concatenate (+)

↳ String Concatenation means add strings together.

↳ The + operator adds the multiple strings together.

eg:-
```
a = "Hello"
b = "Mehndi"
Print (a+b)
```
✓
Output
Hello Mehndi

eg:-
```
a = "Hello"
b = 20
Print (a+b)
```
✗
output
error (Because a is string value and b is int value)

2) List :- [ ]

↳ Python lists are similar to arrays. But the list can contain different types of data. (The items stored in the list are seperated with a comma (,) and enclosed within square brackets [ ].

↳ We can use slice [:] operator to access the data of the list.

↳ Duplicate is allowed.

↳ It is mutable data type.

Syntax:-
list_name = [item1, item2, ... itemn]

eg:- a = [10, 'anu', 10]
Print (a, type(a))

Output
[10, anu, 10], <class : list>

eg:- where you want to print 'Mehndi' only then:-

a = [10, 'Mehndi', True, 10.6, 10]
Print (a[1])

Output
Mehndi

* List slicing:- [:]

eg:- a = [10, 'Mehndi', True, 10.6, 'anu', 10]
Print (a[1:4])

Output
['Mehndi', True, 10.6]

↳ Slicing में एक extra digit लिखते हैं because slicing '-1' से start होती है।

* Index () Method function:-

eg:- a = [10, 'Mehndi', True, 10.6, 'anu', 10]
Print (a.index ('anu'))

Output
4

(1) #List

↳ List is a mutable datatype.

↳ It is similars to array but list contain different type of ~~data type~~ data.

↳ List is a ~~hetro~~ heterogeneous. it is the most used datatypes in Python.

3) Tuple:-

A tuple is similar to the list in many ways. Like lists, tuples also contain the collection of items of diff^n data types.

The items of the tuple are separated with a comma (,) and enclosed in parenthesis ().

① Tuple is a immutable data type.

↳ A tuple is a read-only data structure as we can't modify the size and value of items of a tuple.

Syntax:- tuple_name = (item1, item2, ... itemn)

↳ Duplication are allowed.

↳ eg:- t = (5, 'program', 1+3j)

Q:- t = (10, 20, 'Hello')
Print (t, type(t))

output
(10, 20, 'Hello') <class 'tuple'>

Dictionary:-

↳ Dictionary is an unordered set of a key-value pair of items. It is like a associative array or a hash table where each key stores a specific value.

③ key hold any primitive data type, whereas value is an arbitrary Python Object.

① It is mutable data type

④ The items in the dictionary are separated with the comma (,) and enclosed in the curly Braces {}.

Syntax:- dict_name = {key : value}

Note:- 1) Indexing & Slicing not work.

2) Insertion order is preserved.

3) Heterogeneous elements are allowed.

4) Mutable in nature.

5) Key must be unique, but ~~value~~ duplicate value are allowed.

eg:- a = {'age': 21}
Print(a, type(a))

Output
{'age': 21} <class 'dict'>

eg:- a = {"Name": "Mehndi", "age": 21, "Username": "Zoey"}
Print(a, type(a))

Output
{'Name': 'Mehndi', 'age': 21, 'Username': 'Zoey'}

eg:- a = {"Name": "Mehndi", "age": 21, "Name": "Mehndi"}
Print(a, type(a))

Output
Error (Because Key can't be duplicate)

जहां यहां Name change कर देंगे तो value change हो जाएगा।

Set

1) Set is immutable and iterable.
2) Set is the unordered collection of the data type.
3) It has unique elements. (no duplication)
4) The items of the sets are separated with comma and enclosed in curly braces.

Syntax:- Set_name = {item1, item2, ---- itemn}

Note:- 1) We write the items of set inside the curly braces "{}".
2) Insertion order is not preserved → (Output हमेशा अलग-अलग ही आता है)
3) Indexing & Slicing not work.
4) Heterogeneous elements are allowed.

Set duplicate element को remove कर देता है।
allow not करता है।

eg:- a = {23, 56.7, 'AKKi'}
Print (a, type(a))

Output
{56.7, 23, 'AKKi'}
<class 'set'>
↳ Because insertion order is not preserved.

User input & type casting in Python:-

Program:-
eg:- a = int (input ("Enter value 1 : "))
b = int (input ("Enter value : "))
Print (a+b)

Output
Enter value 1:-10
Enter value 2:-10
20

eg:- eval :-
↳ eval function लिखने के बाद हमें बार-बार int, float etc लिखने की need नहीं पड़ती है।

eg:- a = eval (input ("Enter value 1: "))
b = eval (input ("Enter value 2: "))
Print (a+b)
Print (a-b)

Output
Enter value 1 : 20
Enter value 2 : 10
30
10

String Indexing and Slicing :—

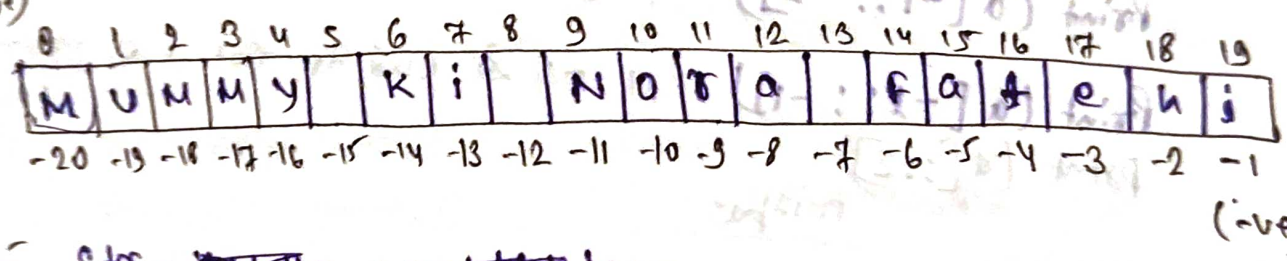
String Indexing :—

↳ Indexing is used to access a single character of a string at a time. It starts from '0' for the first character and increases by 1 for each subsequent character.

↳ To access a character at a particular index, you can use square brackets '[]' with the index number.

↳ We can access characters in a string in two ways :—

1) Accessing characters by positive index number (start from 0)
2) Accessing characters by negative index number (start from -1)



Eg:- Str = ~~"Hello world"~~ "Mummy Ki Nora Fatehi"

```
Print (Str[0])
Print (Str[9])
Print (Str[14])
Print (Str[-1])
Print (Str[-18])
```

Output

M
N
F
i
M

Slicing in String (:)

↳ Slicing is used to access a substring of a string. (TOPIC)

↳ It allows you to extract a part of a string. (TOPIC)

↳ Slicing works with the help of colon ':' operator.

↳ Syntax:-

'string [start_index : end_index : step]'

eg:- a = 'Hello Mehudi'

~~Print (a[2])~~

Print (a[0:7]) // 0 से 6 तक ही।

Print (a[0::2]) // 0 से, end तक, But 2 का increment (1 gap)

Print (a[-1::-2]) // -1 से end तक, But -2 का decrement (1 gap)

Print (a[-1::-1]) // -1 से end तक, But -1 का decrement. (1 उल्टा)

Print (a[::2])

output

Hello M (0:7)

HlOMhd (0:2)

ine ~~Hel~~le (-1::-2)

iduheM olleH

Operator

↳ Operators are used to perform operations on Variables and values. / operands.

↳ Operator is a symbol that operates on a value or a variable.

Types of operator.

1) Arithmetic operator.
2) Assignment "
3) comparison / Relational "
4) Logical "
5) ~~Identity~~ Equality "
6) Bitwise "
7) Ternary operator
8) Shift "
9) Special operator.
   - identity operator
   - membership operator.

1) Arithmetic

↳ Arithmetic Operators are used to perform arithmetic operations on the operands.

Operator	Name	Example
+	Addition	a+b
−	Subtraction	a-b
*	Multiplication	a*b
/	Division	a/b
%	Modulus	a%b
**	Exponentiation	x**y (eg: 3**3 = 27 ; 2**2 = 4)
//	Floor division operator	x//y (eg: x=15, y=2, Print(x//y) Output 7 (nearest whole number))

2) ASSIGNMENT operator :— It is used to assign values to variables.

Operator	Name	Example
=	Assign	x = 5
+=	Add and assign	x += 3
-=	Subtract and assign	x -= 3
*=	Multiply and assign	x *= 3
/=	Divide and assign	x /= 3
%=	Modulus and assign	x %= 3

3) Comparison Operators :— It is used to compare two values.

Operator	Name	Example
==	Equal to	x == y
!=	not equal to	x != y
>	greater than	x > y
<	less than	x < y
>=	greater than or equal to	x >= y
<=	less than or equal to	x <= y

} True/false in answer mostly.

4) Logical Operator :— It is used to combine conditional statements.

Operator	Name	example
AND	If both statements are true, it returns True	eg: x = 5 Print(x > 3 and x < 10) (True)
OR	If one of the statements is true, it returns True	eg:- x = 5 Print(x > 3 or x < 4) (True)
NOT	The output will be inversed.	eg:- x = 5 Print(not(x > 3 and x < 10))

output

False (True if opposite)

5) **Membership operator:** — It is used to test if a sequence is presented in an object.

Operator	Name	example
in	returns True if a sequence with the specified value is present in the object.	x in y eg:- x = ["apple", "banana"] Print ("banana" in x) Output: True
not in	Return True if a sequence with the specified value is not present in the object.	Syntax: x not in y eg:- x = ["apple", "banana"] Print ("Pear" not in x) Output: True

6) **Identity Operator:** —

↳ It is used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location.

Operator	Description	Example
is	returns True if both variables are the same object.	Syntax:- x is y eg:- x = 10 y = 10 Print (x is y) | Output: True
is not	return True if both variables are not the same object.	Syntax:- x is not y Print (x is not y) | output: False

7) **Bitwise operators:** — It is used to compare (binary) numbers.

Operator	Name & Description	example
&	AND → Sets each bit to 1 if both bits are 1.	x & y
|	OR → Sets each bit to 1 if one of two bits is 1.	x | y

operator Precedence

- In Python, operator Precedence determines the order in which operators are evaluated in an expression.
- It defines which operators are evaluated first and which one are evaluated later.
- Here is the order of precedence for operators (from highest to lowest). mann is left to right

Trick :- PEMDAS (see chatgpt)

i) Parentheses ().
ii) Exponentiation (**).
iii) Multiplication (*).
iv) Division (/).
v) Addition (+)
vi) Subtraction(-) etc.
vii) Comparison operators.
viii) logical operator.

eg:-

```
result = (10 + 20 * 30)
print (result)
```

(+ → lower precedence; * → higher precedence)

Output

610 # This is calculated as (10+(20*30)) and not as (10+20)*30.

eg:-

```
result = 2+3*4
Print (result)
```

output

14

```
result = (10+20*30)
print(result)
```

Control Structure in Python

Control Structure

1) Selection Statement / Decision-making Statements
   - 1) if Statement
   - 2) if-else statement
   - 3) if-elif-else statement
2) Looping / Iterative Statements
   - while
   - for
3) transfer Statements
   - Break
   - continue
   - Pass

Selection or Conditional Branching

1) ~~Decision making~~ if statement :-

1) if - Statement

↳ In Python, it will go inside the Block only, if the condition is true otherwise, it will be not execute the Block.

expression :-

```
if [Conditional expression]:
    [Statement (s) to execute]
```

or

```
if Condition:
    Statement
```

eg:- i)
```
a = 10
if a%2 == 0:
    Print("Even no")
```

user ka input store

ii)
```
a = int(input("Enter the value of a"))
if a%2 == 0:
    Print(a, "is even no.")
```

2) if - else Statement

↳ if the condition is true then it will execute the if block otherwise, it will execute the else block.

expression:-

```
if condition :
    statement
else :
    statement
```

eg:- i)

```
a = 10
if a % 2 == 0 :
    Print ("Even no.")
else :
    Print ("odd no.")
```

output

even no.

ii)

```
a = int (input ("Enter the value "))
if a % 2 == 0 :
    Print (a, "Even no")
else :
    Print (a, "odd no.")
```

output

Enter the value
10, even no.

3) if - elif - else Statement.

↳ if the condition is true, then it will execute the if Block otherwise, it will execute the elif Block. Again if the condition is ~~elif~~ not met, then it will move to the else Block.

expression:-

```
if condition :
    Statement
elif condition :
    Statement
else :
    Statement
```

eg:- (i)

```
a=10
if a<10:
    Print ("The statement is true")
elif a==10:
    Print ("The statement is [redacted] match")
else :
    Print ("The statement is false")
```

eg:- (ii)

```
Per = 55 'Or' Per=int (input ("Enter the value"))
if Per=>60:
    Print ("first Div")
elif Per >= 48:
    Print ("second Div")
elif Per > 35:
    Print ("Third Div")
else:
    Print ("Fail)
```

2) Looping Statement / Iterative Statement

↳ Range

* range (5) → By default starts with 0 (zero).
  - # Start = 0
  - # Condition < 5
  - # increment 1

Output
0
1
2
3
4

Range() function used to generate the sequence of numbers according to given condition

* range (1, 6) → 1 is start because it is already mentioned.
  - # Start = 1
  - # Condition < 6
  - # increment 1

Output
1 4
2 5
3

* range (1, 6, 2)
  - # Start = 1
  - # Condition < 6
  - # increment 2

Output
1
3
5

i) for loop.

↳ (initialization, condition, iteration)
↳ It executes the code until condition is false.
↳ It is used when number of iterations are known in advance/already.

eg:- i) for n in range (5):
    Print (n)

Output
0 3
1 4
2

ii) for n in range (1, 6):
    Print (n)

Output
1 3 5
2 4

iii) for n in range (1, 6, 2):
    Print (n)

Output
1 3 5

for a in range (1, 11):

print("2 * ", a, " = " 2 * a) or print (2 * a)

Output

2 * 1 = 2

2 * 2 = 4

2 * 3 = 6

2 * 4 = 8

2 * 5 = 10

2 * 6 = 12

2 * 7 = 14

2 * 8 = 16

2 * 9 = 18

2 * 10 = 20

for i in range (10, 0, -1):
print(i)

Output

10
9
8
7
6
5
4
3
2
1

for i in range (10, 0, -2):
print(i)

Output

10
8
6
4
2

11) While loop :

↳ While loop executes till the condition becomes false

↳ (start, condition, iteration)

↳ (increment/decrement)

↳ (range नहीं लिखते हैं) .

↳ i++ की जगह पर i+1 का भी प्रयोग कर सकते हैं ।

Q1) i=1
while i<=10:
    Print(i, "Mehudi")
    i = i+1
    Print(i)

Output
Mehudi
11 (Print(i) का output है)

Q2) a=10
while a>=1:
    Print("Vrg")
    a = a-1
    Print(a)

Output
Vrg
10 times vrg
0 (Print(a) का output है)

Q3) B=1
while B<=10 :
    Print(a)
    a = a+1

Output
1
2
3
4
5
6
7
8
9
10

Transfer ~~Jumping~~ Statements

1) Break Statement

↳ It is used to terminate the execution of the current loop ~~or switch~~ the statements.

eg:- 1)
```
for num in range (1,11●):
    if num == 4:
        break
    else:
        Print(num)
```

Output
1
2
3
4

2) Continue Statement.

↳ The continue statement doesn't break the loop, ~~when~~ it skips the specific part of the loop and jumps to the next iteration of the loop immediately.

eg:- 1)
```
for num in range (1, 10):
    if num == 4:
        continue
    else:
        Print (num)
```

Output
1
2
3
5
6
7
8
9
10