12/09/25

O/P :- invalid syntax.

Day - 4

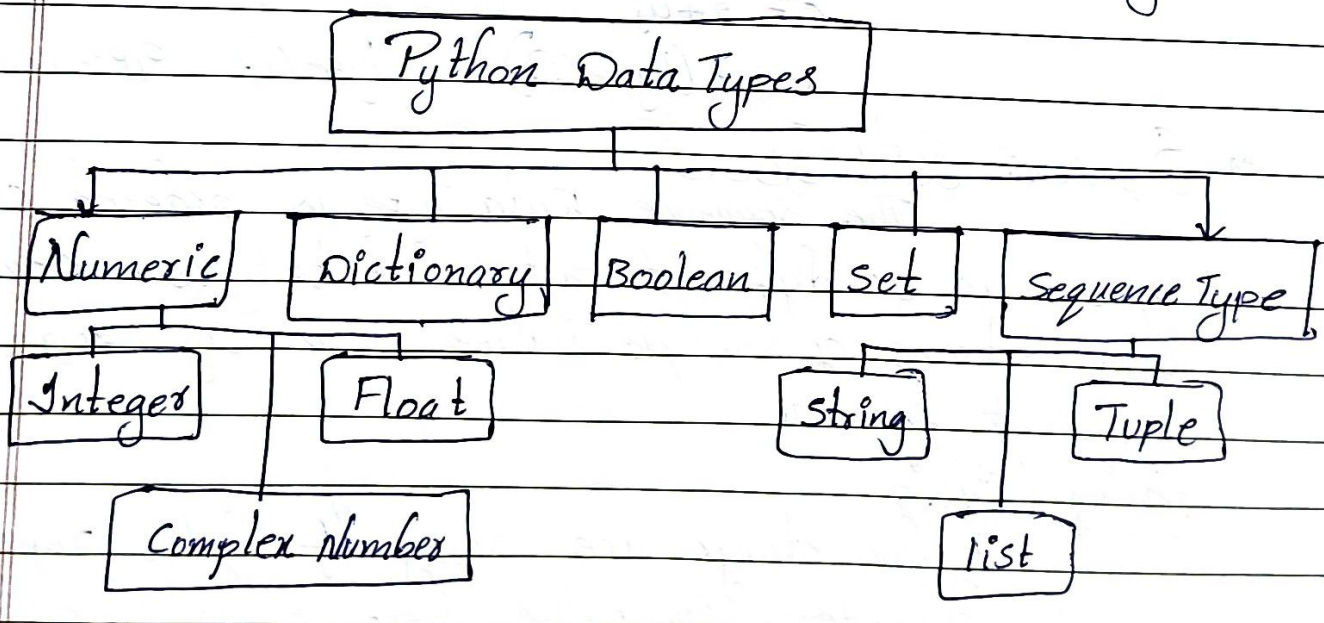
Data Types :-

Data types in python are classification or categorization of data items.

It represents kind of value that tells what operations can be performed on a particular data, a variable can hold.

Python data types are classes and variables are instances (objectives) of these classes.

→ Python understands the datatype automatically.



Numeric Datatypes :-

Python numbers represent data that has a numeric value. A numeric value can be an integer, a floating number or Even a Complex number.

These values are defined as int, float and Complex classes.

is that tuples are immutable. Tuples cannot be modified after it is created.

Creating a Tuple in Python:-

In Python, tuples are created by placing a sequence of values separated by a 'comma' with or without the use of parentheses for grouping data sequence.

Tuples can contain any number of elements and of any datatype (like strings, integers, lists, etc.).

Ex:-

```
#initiate Empty Tuple
tup1 = ()
tup2 = ('Greeks', 'For')
Print = ("\n Tuple with the use of string:",
                                    tup2)
```

Access Tuple items:-

In order to access tuple items refer to the index number. Use the index operator [ ] to access an item in a tuple.

```
tup1 = tuple([1,2,3,4,5])
print (tup1[0])
print (tup1[-3])
```

Boolean Data Type in Python:-

It is one of the two built-in values, True or False. Boolean objects that are equal to True are truthy (true) and those equal to false are falsy (False). However non-Boolean objects can be evaluated in a Boolean context as well and determined to be true or false. It is denoted by class bool.

Ex:- print (type (True)) o/p = Bool

print (type (False)) o/p = Bool

print (type (true)) Error, because, true is not a valid keyword.

Set Data Type:-

Set is an unordered collection of data types that is iterable, mutable, and has no duplicate elements. The order of elements in a set is undefined though it may consist of various elements.

Create a set in Python:

Sets can be created by using the built-in set() function with an iterable object or a sequence by placing the sequence inside curly braces, seperated by a 'Comma'. The type of elements in a set need not be the same, various mixed-up data type values can also be passed to the set.

Ex:-

```
S1 = set()
S1 = set("GeeksForGeeks")
print("Set with the use of string:", S1)
S2 = set(["Geeks", "For", "Geeks"])
print("Set with the use of list:", S2)
```

o/p:-

Set with the use of string: {'S', 'o', 'F', 'e', 'k', 's'}
Set with the use of list : {'Geeks', 'For'}

Access set items:-

Set items can't be accessed by referring to an index, since sets are unordered the items have no index. But we can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in the keyword.

Ex:-

```
set1 = set(["Geeks", "For", "Geeks"])
print(set1)
# loop through set
for i in set1:
    print(i, end=" ")
# check if item Exist in set
print("Geeks" in set1)
```

Dictionary Data Type :-

It is a collection of data values, used to store data values like a map, unlike other python Data Types, a dictionary holds a key : value pair. key-value is provided in dictionary to make it more optimized. Each key-value pair in a dictionary is separated by a colon :, whereas each key is separated by a 'comma'.

Create a Dictionary in python :-

Values in a dictionary can be of any datatype and can be duplicated, whereas keys can't be repeated and must be immutable.

The dictionary can also be created by the built in fun^c dict().

⇒ Dictionary keys are case sensitive, the same name but different cases of key will be treated distinctly.

```
Ex:- d = {}
     d = {1: "Geeks", 2: 'For', 3: 'Geeks'}
     print (d)
     d1 = dict ({1: 'Geeks', 2: 'For', 3: 'Geeks'})
     print (d1).
```

Accessing key-value in Dictionary :-

In order to access items of a dictionary refer to its key value.name. key can be used inside square brackets. using get() method we can access dictionary elements.

```
Ex:-
     d = {1: 'Geeks', 'name': 'For', 3: 'Geeks'}
# Accessing an element using key.
     print (d['name'])
# Accessing a element using get
     print (d.get(3))
```

O/P :- For
Geeks.

Advantages and Disadvantages of python :-

Advantages :-

→ Easy to learn & Readable :- Simple syntax, close to English, makes it beginner-friendly.

→ Versatile & Multi-Purpose :- Used in web development, data science, AI/ML, automation, IoT, game development, etc.

→ Large Community & Libraries :- Huge ecosystem of libraries (Numpy, Pandas, TensorFlow, Django, Flask, etc.)

→ Cross-Platform :- Works on windows, Mac, Linux, and many embedded systems.

→ Rapid Development :- Faster prototyping and development due to simplicity and built-in functions.

→ Integration :- Can integrate with other languages (C, C++, Java) and databases easily.

→ Strong Support For AI & Data Science :-
Python is the most popular choice for machine learning, data analysis, and deep learning.

Disadvantages :-

→ Slower Execution Speed :- Interpreted language → slower than Compiled languages like C++ or Java.

→ High Memory Usage :- Not ideal for memory-intensive tasks (like mobile apps or Embedded systems).

→ Not Great for Mobile Development :- Limited use in mobile app development Compared to java/kotlin (Android) or swift (ios).

→ Weak in Multithreading :- Global Interpreter Lock (GIL) prevents true parallel execution of threads.

→ Runtime Errors :- Being dynamically typed, errors (like type mismatches) often appear at runtime, not compile-time.

→ Database Access Limitations :- slower and less robust Compared to languages like java when working with heavy enterprise databases.

Types of Variables:-

Local Variables:-

Defined inside a function and can only be used within that function.

Ex:-
```
def my_function():
    message = "Hello" #local variable.
    print(message)
my_function()
```

Global Variables:-

Defined outside of all functions and can be used anywhere in the program.

Ex:-
```
name = "Priya" #global variable
def greet():
    print ("Hello", name) #accessing global variable.
greet()
print(name)
```

If you want to modify a global variable inside a function, use global keyword.

Ex:-
```
Count = 0
def increment():
    global Count
    Count += 1
increment()
print(Count) #1
```

Instance Variables:-

Variables that belong to each object (instance) of a class.

They are defined inside a Constructor. (__init__) using self.

Ex:-
```
class student:
    def __init__(self, name, age):
        self.name = name #instance
        self.age = age #instance
s1 = student("Priya", 22)
s2 = student("Priya", 23)
```

Print (s1.name) # Priya
print (s2.name) # Priya

Class Variables:-

Variables that are shared among all objects of a class. They are defined inside the class but outside methods.

Ex:- Class student:
    school = "ABC school" # class variable (shared)
    def __init__(self, name):
        self.name = name # instance variable (unique)
s1 = student ("Priya")
s2 = Student ("Ravi")
Print (s1.school) # ABC school
print (s2.school) # ABC school

Operators:-

In Python Programming, Operators in general are used to perform operations on values and variables. These are standard symbols used for logical and arithmetic operations. In this article, we will look into diff types of Python Operator.

Operators:- These are the special symbols. Eg:- +, -, *

Operand:- It is the value on which the operand is applied.

Types of Operators in Python:-

Arithmetic Operator :- +, -, *, /, %, // **-Exponention
Relational Operator :- <, <=, >=, >, ==, !=
Logical operator :- AND, OR, NOT
Bitwise Operator :- &, |, <<, >>, ~, ^
Assignment Operator :- =, +=, -=, *=, %=

Arithmetic :- a = 15, b = 4, a+b a-b a*b a%b

Relational / Comparision Operator :-

It Compares the values. It either returns True or False according to Condition.

Ex:- a=13, b=33

print (a>b) print (a==b)
|| (a<b) || (a!=b)
|| (a<=b) || (a>=b)

Logical Operators :-

It perform Logical AND, Logical OR and Logical NOT operations. It is used to Combine Conditional statements.

Ex:- a = True print (a and b)
b = False print (a or b)
print (not a)

X	Y	and	or	not	not
T	T	T	T	F	F
T	F	F	T	F	T
F	T	F	T	T	F
F	F	F	F	T	T

Bitwise Operators :-

It act on bits and perform bit-by-bit operations. These are used to operate on binary num.

-> Bitwise NOT -> ~
-> Bitwise shift -> << >>
-> Bitwise and -> &
-> Bitwise XOR -> ^
-> Bitwise OR -> |

Ex:- a=10 print (a&b)
b=11 print (a|b)
print (~a)
print (a^b)
print (a>>2)
print (a<<2)

identity operators :-

In python, is and is not are the ident

operators, both are used to check if two values are located on the same part of the memory. Two variables that are Equal do not imply that they are identical.

is — True if the operands are identical.

is not — True if the operands are not identical.

Ex:- a=10 print (a is not b) True

b=20 print (a is c) True

c=a

Membership Operators:-

In Python, in and not in are the membership operators, that are used to test whether a value or variable is in a sequence.

in — True if value is found in the sequence.

not in — True if value is not found in the sequence.

Ex:- x=24

y=20

list = [10, 20, 30, 40, 50]

```
if (x not in list):
    print ("x is NOT present in given list")
else:
    print ("x is present in given list")
if (y in list):
    print ("y is present in given list")
else:
    print ("y is not present in given list")
```

Ternary Operator:-

Ternary Operators also known as Conditional Expressions are operators that evaluate something based on a condition being true or false.

It simply allows testing a condition in a single-line replacing the multiline if-else making the code compact.

syntax:- [on true] if (Expression) else [on false]