Fundamentals in the Python Language

What are Tokens in Python?

In Python, **tokens** are the **smallest units or building blocks of a Python program**. They are the basic elements that the Python interpreter recognises for syntax and semantic analysis.
Whenever a Python program is executed, it is broken down into **tokens** by the interpreter.

Types of Tokens in Python
Python has **five types of tokens**:

Token Type	Description	Examples
1. **Keywords**	Reserved words with special meaning in Python	if, else, while, def, return, etc.
2. **Identifiers**	Names used to identify variables, functions, classes	x, sum, my_function, StudentName
3. **Literals**	Data values assigned to variables	123, 3.14, 'hello', True, None
4. **Operators**	Symbols that perform operations on variables or values	+, -, *, /, ==, and, or, etc.
5. **Punctuators (Separators)**	Symbols used to structure code or separate tokens	(,), :, ,, ., [ ], { }, =, etc.

1. Keywords
- Predefined by Python and cannot be used as identifiers.
- Python has **35 keywords** (as of Python 3.11), like:

if, else, elif, for, while, def, class, return, import, pass, break, continue, try, except, finally, with, yield, True, False, None, and, or, not, in, is

2. Identifiers
- The **names** given to variables, functions, classes, etc.
- Rules:
  - Must begin with a letter (A–Z/a–z) or underscore _
  - Followed by letters, digits (0–9), or underscores
  - Cannot be a keyword
  - Case-sensitive (Sum and sum are different)

Examples:
name = "Alice"
total_marks = 90
def calculate(): ...

3. Literals
- These are **fixed values** used in Python.
- Types of literals:
  - **Numeric**: int, float, complex
  - **String**: 'hello', "world", '''multi-line'''
  - **Boolean**: True, False
  - **Special**: None
  - **Collection**: Lists [1, 2], Tuples (3, 4), Sets {5, 6}, Dictionaries {"key": "value"}

Operators

- Perform **arithmetic, logical, relational, and bitwise** operations.

Type	Examples
Arithmetic	+, -, *, /, //, %, **
Relational/Comparison	==, !=, >, <, >=, <=
Logical	and, or, not
Assignment	=, +=, -=, *= etc.
Bitwise	&, `
Membership	in, not in
Identity	is, is not

5. Punctuators / Separators

These are **symbols** used to structure the code:

```
( )    # Parentheses
[ ]    # Square brackets
{ }    # Curly braces
:      # Colon
,      # Comma
.      # Dot
;      # Semicolon (rarely used)
=      # Assignment
@      # Decorator symbol
->     # Function return type hint
```

Example program to demonstrate all the tokens used in the code:

Sample program to accept the radius of a circle from the user, and print the area after computation.

```
PI = 3.14159

# Taking input from the user
radius = float(input("Enter radius: "))

# Computing the area
area = PI * radius ** 2

# Displaying the output
print("Area of Circle is:", result)
```

Token Type	Examples from the code
Keywords	import, def, return
Identifiers	PI, radius, area (Variables) print, input, float (Functions)
Literals	3.14159, 2 (Numeric literals) "Enter radius: ", "Area of Circle is:" (String literals)
Operators	=, *, **
Punctuators	( ) and ,
Comments	# ……..

Python Operators with Examples

1. Arithmetic Operators - Used for mathematical operations.

Operator	Description	Example	Result
+	Addition	10 + 5	15
-	Subtraction	10 - 5	5
*	Multiplication	10 * 5	50
/	Floating pt Division	10 / 4	2.5
//	Integer Division	10 // 4	2
%	Modulus (remainder)	10 % 4	2
**	Exponentiation	2 ** 3	8

2. Assignment Operators - Used to assign values to variables.

Operator	Example	Equivalent To
=	x = 5	Assign value
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 2	x = x * 2
/=	x /= 2	x = x / 2
//=	x //= 2	x = x // 2
%=	x %= 2	x = x % 2
**=	x **= 3	x = x ** 3

3. Comparison (Relational) Operators - Used to compare values and return True or False.

Operator	Description	Example	Result
==	Equal to	5 == 5	True
!=	Not equal to	5 != 3	True
>	Greater than	5 > 3	True
<	Less than	3 < 5	True
>=	Greater than or equal	5 >= 5	True
<=	Less than or equal	4 <= 5	True

4. Logical Operators - Used to combine conditional statements.

Operator	Description	Example	Result
and	True if both are true	True and True	True
or	True if at least one is true	True or False	True
not	Reverses result	not True	False

5. Bitwise Operators - Operate on binary representations.

Operator	Description	Example	Result
&	AND	5 & 3 → 101 & 011 = 001	1
`	`	OR	`5
^	XOR	5 ^ 3 → 101 ^ 011 = 110	6
~	NOT	~5 (inverts bits)	-6
<<	Left Shift	5 << 1 = 10	10
>>	Right Shift	5 >> 1 = 2	2

6. Membership Operators - Check membership in a sequence (like list, string, etc.)

Operator	Description	Example	Result
in	True if value exists	'a' in 'apple'	True
not in	True if value not exists	'x' not in 'apple'	True

7. Identity Operators - Check if two variables refer to the same object.

Operator	Description	Example	Result
is	True if same object	x is y	True/False
is not	True if not same object	x is not y	True/False

Problems Using Arithmetic Operators (Only Questions)
Each program will take input from the user, as mentioned in the input section, and provide the output accordingly. Use appropriate formula for each task.

1. **Calculate the area of a rectangle**
   - Input: Length and width
   - Output: Area
2. **Find the perimeter of a square**
   - Input: Side length
   - Output: Perimeter
3. **Convert temperature from Celsius to Fahrenheit**
   - Formula: $F = (C \times 9/5) + 32$
   - Input: Temperature in Celsius
   - Output: Temperature in Fahrenheit
4. **Calculate the average of three numbers**
   - Input: Three numbers
   - Output: Average
5. **Find the simple interest**
   - Formula: $SI = (P \times R \times T) / 100$
   - Input: Principal, Rate, Time
   - Output: Simple Interest
6. **Swap two numbers without using a third variable**
   - Use arithmetic operations only
7. **Calculate the area and circumference of a circle**
   - Input: Radius
   - Output: Area and Circumference
   - Use $\pi = 3.14$
8. **Calculate the number of minutes and seconds from the total seconds**
   - Input: Total seconds
   - Output: Minutes and remaining seconds
9. **Find the cube and square of a number**
   - Input: Any number
   - Output: Square and Cube
10. **Calculate the speed of a vehicle**
    - Formula: Speed = Distance / Time
    - Input: Distance and Time
    - Output: Speed