Data Types, Operators and Expressions

Data Types in Python

- Data type:
  - "A classification of data that determines the type of value it can hold"
  - Determines the amount of memory it takes, and the operations that can be performed on it.
- Numeric Data Types: int, float, complex
- Boolean
- String

Operators and Expressions

- **Operators**: Special symbols or keywords used to perform operations on variables and values.
- **Expression**: Combination of values, variables, and operators that can be evaluated to produce a result.

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	3 + 2	5
-	Subtraction	5 - 1	4
*	Multiplication	4 * 2	8
/	Division	10 / 2	5
//	Floor Division	10 // 3	3
%	Modulus (Remainder)	10 % 3	1
**	Exponentiation	2 ** 3	8

Relational/Comparison Operators

Operator	Description	Example	Result
==	Equal to	5 == 5	TRUE
!=	Not equal to	3 != 2	TRUE
>	Greater than	4 > 1	TRUE
<	Less than	2 < 3	TRUE
>=	Greater or equal	3 >= 3	TRUE
<=	Less or equal	4 <= 5	TRUE

Logical Operators

Operator	Description	Example	Result
and	True if both are True	True and False	False
or	True if at least one is True	True or False	True
not	Inverts truth value	not True	False

Assignment Operators

Operator	Example	Same As
=	x = 5	assign 5 to x
+=	x += 2	x = x + 2
-=	x -= 1	x = x - 1
*=	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

String Operators

Operator	Description	Example	Result
+	Concatenation	"Hi" + "John"	"HiJohn"
*	Repetition	"Ha" * 3	"HaHaHa"

Membership Operators

Operator	Description	Example	Result
in	True if found	'a' in 'apple'	TRUE
not in	True if not found	'x' not in 'apple'	TRUE

Bitwise Operators

Operator	Description	Example	Result
&	Bitwise AND	5 & 3	1
|	Bitwise OR	5 | 3	7
^	Bitwise XOR	5 ^ 3	6
~	Bitwise NOT	~5	-6
<<	Left Shift	5 << 1	10
>>	Right Shift	10 >> 1	5

Unary Operators

Operator	Description	Example	Result
+x	Unary Plus (returns x as is)	x = -5; y = +x	-5
-x	Unary Minus (negates x)	x = 5; y = -x	-5
~x	Bitwise NOT (flips all bits of x)	x = 5; y = ~x	-6

Expressions

```
x = 5
y = 3

z = x + y * 2
# Expression equivalent to
# 5 + 3*2 = 11

is_even = (x % 2 == 0)
# is even is now equal to False


age = 20
citizen = True
can_vote = (age >= 18) and (citizen == True)
```

Operator Precedence

- Rules for specifying the order in which the operators in an expression must be evaluated when the expression has several operators
- When two operators share an operand, the operator with the higher precedence goes first.
- Ex: multiplication has higher precedence than addition
  - a + b * c is treated as a + (b * c)
- Precedence rules can be overridden by explicit parentheses

Associativity

- When two operators share an operand and the operators have the same precedence, then the expression is evaluated according to the associativity of the operators.
- Associativity is specified as
  - left to right
  - right to left
- Examples:
  - the ** operator has right-to-left associativity, hence a ** b ** c is treated as a ** (b ** c)
  - the / operator has left to right associativity, hence a / b / c is treated as (a / b) / c.

Operator Precedence and Associativity

Operators	Description	Associativity
()	Parentheses (grouping)	N/A
**	Exponentiation	Right to Left
+x, -x, ~x	Unary plus, minus, bitwise NOT	Right to Left
*, /, //, %	Multiplication, division, floor division, modulus	Left to Right
+, -	Addition and subtraction	Left to Right
<<, >>	Bitwise shift left and right	Left to Right
&, ^, |	Bitwise AND, XOR, OR	Left to Right
==, !=, >, <, >=, <=, in, not in	Comparisons, identity, membership	Left to Right
not	Logical NOT	Right to Left
and, or	Logical AND, OR	Left to Right
=, +=, -=, *=, /=, //=, %=, **=, &=, |=, ^=, >>=, <<=`	Assignment operators	Right to Left