Python Programming

Akshita Chanchlani

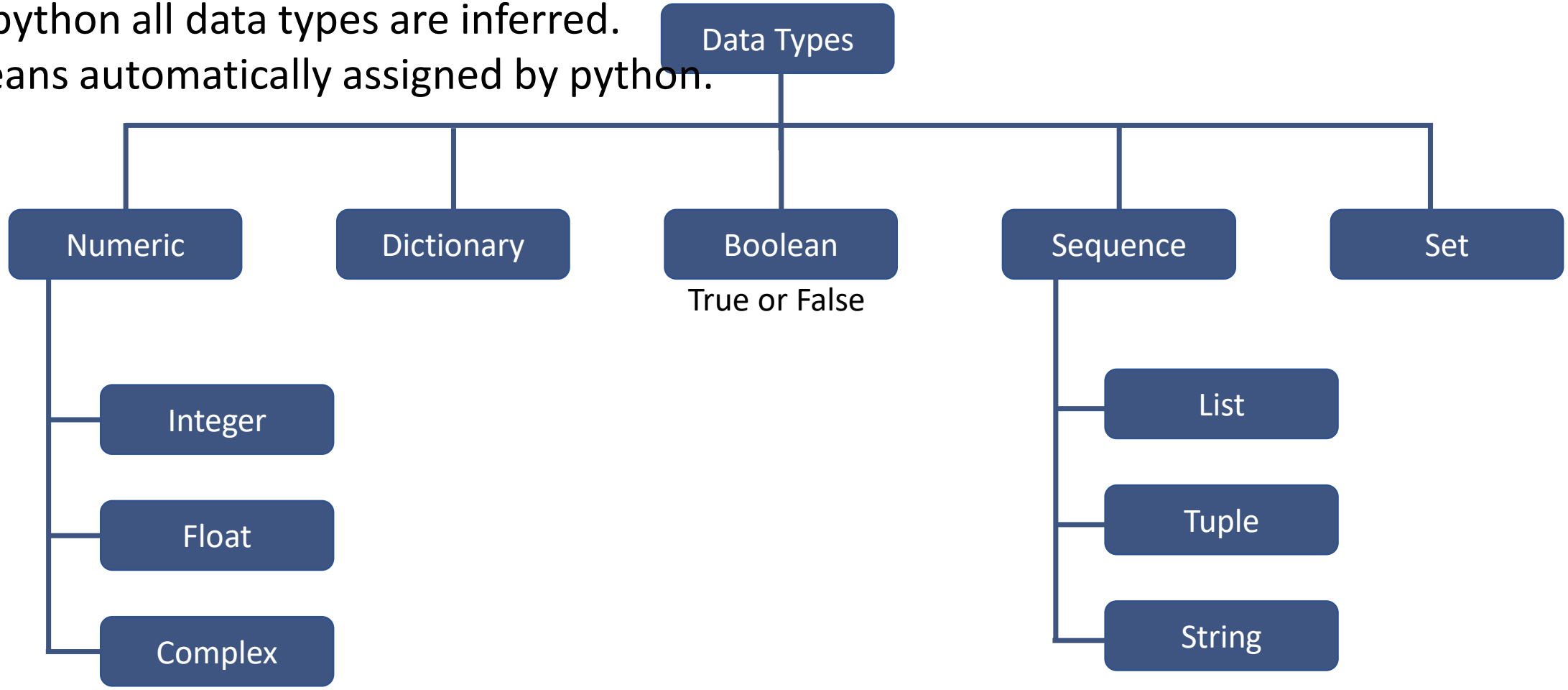
Data Types

Need of Data Types

- Data type is an attribute associated with a piece of data that tells a computer system how to interpret its value
- Understanding data types ensures that data is collected in the preferred format and the value of each property is as expected
- It helps to know what kind of operations are often performed on a variable

Python Data Types

- In python all data types are inferred.
- Means automatically assigned by python.



Literals

- Literal is a raw data given in a variable or constant

- **Numeric Literals**
  - Numeric Literals are immutable (unchangeable)
  - Numeric literals can belong to 3 different numerical types:
    - Integer (value = 100)
    - Float (salary = 15.50)
    - Complex (x = 10 + 5j)

- **String literals**
  - A string literal is a sequence of characters surrounded by quotes
  - We can use both single, double, or triple quotes for a string
  - E.g., name = “steve” or name = ‘steve’

Literals

- **Boolean literals**
  - A Boolean literal can have any of the two values: True or False
  - E.g., can_vote = True or can_vote = False

- **Special literal**
  - Python contains one special literal i.e. None
  - We use it to specify that the field has not been created
  - E.g., value = None

- **Literal Collections**
  - Used to declare variables of collection types
  - There are four different literal collections List literals, Tuple literals, Dict literals, and Set literals

Type Hinting

- Due to the dynamic nature of Python, inferring or checking the type of an object being used is especially hard
- Data types are dynamically assigned.
- This fact makes it hard for developers to understand what exactly is going on in code they haven't written. As a result they resort to trying to infer the type with around 50% success rate.
- **Why type hints?**
  - **Helps type checkers:** By hinting at what type you want the object to be the type checker can easily detect if, for instance, you're passing an object with a type that isn't expected
  - **Helps with documentation:** A third person viewing your code will know what is expected where, ergo, how to use it without getting them TypeErrors
  - **Helps IDEs develop more accurate and robust tools:** Development Environments will be better suited at suggesting appropriate methods when know what type your object is

Type Conversion

- The process of converting the value of one data type (integer, string, float, etc.) to another data type is called type conversion
- Python has two types of type conversion.
  - Implicit Type Conversion
    - Python automatically converts one data type to another data type
    - This process doesn't need any user involvement
    - E.g.
      - result = 10 + 35.50
  - Explicit Type Conversion
    - Users convert the data type of an object to required data type
    - We use the predefined functions like int(), float(), str(), etc to perform explicit type conversion
    - E.g.
      - num_str = str(1024)

Type Conversion

- Type Conversion is the conversion of object from one data type to another data type
- Implicit Type Conversion is automatically performed by the Python interpreter
- Python avoids the loss of data in Implicit Type Conversion
- Explicit Type Conversion is also called Type Casting, the data types of objects are converted using predefined functions by the user
- In Type Casting, loss of data may occur as we enforce the object to a specific data type

Operators

Operators

- Operators are special symbols in Python that carry out arithmetic or logical computation
- The value that the operator operates on is called the operand
- Types
  - Arithmetic operators
  - Comparison operators
  - Logical operators
  - Bitwise operators
  - Special operators
  - Membership operators

Arithmetic Operators

Operator	Meaning	Example
+	Add two operands or unary plus	x + y + 2
-	Subtract right operand from the left or unary minus	x – y
*	Multiply two operands	x * y
/	Divide left operand by the right one (always results into float)	x / y
%	Modulus - remainder of the division of left operand by the right	x % y
//	Floor division - division that results into whole number adjusted to the left in the number line	x // y
**	Exponent - left operand raised to the power of right	x**y

Comparison operators

Operator	Meaning	Example
>	Greater than - True if left operand is greater than the right	x > y
<	Less than - True if left operand is less than the right	x < y
==	Equal to - True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to - True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to - True if left operand is less than or equal to the right	x <= y

Logical operators

Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

Assignment Operators

Operator	Meaning	Example
=	x = 5	x = 5
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5
//=	x //= 5	x = x // 5

Operator	Meaning	Example
**=	x **= 5	x = x ** 5
&=	x &= 5	x = x & 5
|=	x |= 5	x = x | 5
^=	x ^= 5	x = x ^ 5
>>=	x >>= 5	x = x >> 5
<<=	x <<= 5	x = x << 5

Bitwise operators

Operator	Meaning	Example
&	Bitwise AND	x & y = 0 (0000 0000)
|	Bitwise OR	x | y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = 14 (0000 1110)
>>	Bitwise right shift	x >> 2 = 2 (0000 0010)
<<	Bitwise left shift	x << 2 = 40 (0010 1000)

Special operators

Operator	Meaning	Example
is	True if the operands are identical (refer to the same object)	x is True
is not	True if the operands are not identical (do not refer to the same object)	x is not True

Membership Operators

Operator	Meaning	Example
in	True if value/variable is found in the sequence	5 in x
not in	True if value/variable is not found in the sequence	5 not in x