1. What is Data Types?

A. Data types are the classification or categorization of data items. They represent the kind of value that tells what operations can be performed on a particular data. Python has the following standard data types:

1. Integer (whole numbers, e.g., 1, 2, 3)

2. Float (decimal numbers, e.g., 3.14, -0.5)

3. String (sequences of characters, e.g., "hello", 'hello')

4. Boolean (true or false values)

5. List (ordered collections of items, e.g., [1, 2, 3], ["a", "b", "c"])

6. Dictionary (unordered collections of key-value pairs, e.g., {"name": "John", "age": 30})

7. Set (unordered collections of unique items, e.g., {1, 2, 3}, {"a", "b", "c"})

1. Integer: integer is a whole number, either positive, negative, or zero, without a fractional part.

- 2 + 3 = 5 (addition)

- 5 - 2 = 3 (subtraction)

- 4 \* 5 = 20 (multiplication)

- 10 / 2 = 5 (division)

- 10 % 3 = 1 (modulus/remainder)

Checking if a Value is an Integer

Example:

For = ["apple", "banana", "cherry"]  
print(For)

1. Float: float is a data type that represents a numerical value with a fractional part, i.e., a number with a decimal point. Floats are used to represent real numbers,

num = 10.5

a = 9.99

print(type(a))

- 2.5 + 3.5 = 6.0 (addition)

- 4.5 - 2.2 = 2.3 (subtraction)

- 3.0 \* 2.5 = 7.5 (multiplication)

- 10.0 / 2.5 = 4.0 (division)

Example:

X = 10

Y = 20

Print(X+Y)##30

X = 20

Y = 10

Print(X-Y) ##10

1. String: string is a sequence of characters, such as letters, numbers, or symbols, enclosed in quotes (either single quotes ' ' or double quotes " "). Strings are immutable, meaning they cannot be changed after creation.

- hello (a simple string)

- 'hello world' (a string with a space)

- "hello world" (same as above, using double quotes)

- '' (an empty string)

- '123' (a string containing numbers)

- 'abc' (a string containing letters)

- '!@#' (a string containing special characters)

Example:

S = ('hello')(“ GOOD”)”’’ My name

Is Varun’’)

Print(s)

Print(type(s))## string

1. Boolean: Boolean is a data type that represents truth values, which can be either:

- True

- False

- and (logical and): True and False = False

- or (logical or): True or False = True

- not (logical not): not True = False

type(True) = <class 'bool'>

isinstance(x, bool) = True

Example:

x = "Hello"  
y = 11  
  
print(bool(x))# True  
print(bool(y)) # True

1. List: list is a data structure that stores a collection of items, which can be of any data type, including strings, integers, floats, and other lists.

- fruits = ['apple', 'banana', 'cherry'] (a list of strings)

- numbers = [1, 2, 3, 4, 5] (a list of integers)

- mixed = ['hello', 42, 3.14, True] (a list with different data types)

- nested = ['a', 'b', ['c', 'd'], 'e'] (a list with a nested list)

type(fruits) = <class 'list'>

isinstance(x, list) = True

Example :

mylist = ["Ram", "Sita", "Lakshman"]  
print(type(mylist))

1. Dictionary: Dictionary (dict) is a data structure that stores a collection of key-value pairs

- person = {'name': 'John', 'age': 30, 'city': 'New York'}

- colors = {'red': '#FF0000', 'green': '#00FF00', 'blue': '#0000FF'}

type(person) = <class 'dict'>

isinstance(x, dict) = True

Example:

x = {  
  "brand": "Ford",  
  "model": "Benz",  
  "year": 1999  
}  
print(x)

1. Set: set is a collection of unique elements, which can be of any immutable data type (e.g., strings, integers, tuples)

Examples of Sets:

- fruits = {'apple', 'banana', 'cherry'}

- numbers = {1, 2, 3, 4, 5}

- mixed = {'hello', 42, 3.14, ('a', 'b')}

type(fruits) = <class 'set'

isinstance(x, set) = True

Example:

Type = {"Iphone", "Samsung", "Lenovo"}  
  
print(len(Type))

1. What are the operators in python?
2. Python has various types of operators for performing different operations. Here are some of the most commonly used operators in Python:

* Arithmetic Operators
* Comparison Operators
* Logical Operators
* Assignment Operators
* Bitwise Operators
* Membership Operators
* Identity Operators

1. Arithmetic operators: Arithmetic operators are symbols used in programming to perform mathematical operations. In Python, the following are arithmetic operators:

1. Addition: a + b # print ( 10 + 20 )

2. Subtraction: a – b # print ( 10 – 20 )

3. Multiplication: a \* b # print ( 10 \* 20 )

4. Division: a / b # print ( 10 / 20 )

5. Modulus (remainder): a % b # print ( 10 % 20 )

6. Exponentiation: a \*\* b # print ( 10 \*\* 20 )

7. Floor Division: a // b # print ( 10 // 20 )

Example:

X = 10

Y = 10

Print(X+Y)##20

Print(x-y)##0

Print(x\*y)##100

1. Comparison Operators: Comparison operators are used in programming to compare two values or expressions and determine their relationship. In Python, the following are comparison operators:

1. Equal: a == b print ( 10 == 20 )## False

2. Not equal: a != b print ( 10 != 20 )## True

3. Greater than: a > b print ( 10 > 20 )## False

4. Less than: a < b # print ( 10 < 20 ) ## True

5. Greater than or equal: a >= b print ( 10 >= 20) ## False

6. Less than or equal: a <= b print ( 10 <= 20 )## True

7. Identity: a is b

8. Non-identity: a is not b

1. Logical Operators: Logical operators are used to combine conditional statements and evaluate them to produce a boolean result (True or False). In Python, the following are logical operators:

1. And: a and b

2. Or: a or b

3. Not: not a

Examples:

- True and False = False

- True or False = True

- not True = False

- not False = True

Example:

For 1 = ‘’Hello world’’

For 2 = For 1

Print(for 1 and for 2)## True

Print(for 1 or for 2)## True

Print(Not For 1)## False

1. Assignment Operators: Assignment operators are used to assign a value to a variable or attribute in Python. The basic assignment operator is the equals sign (=), which assigns the value on the right-hand side to the variable or attribute on the left-hand side.

a = 5

a += 3

a -= 2

a \*= 4

a /= 2

a %= 3

a \*\*= 2

a //= 2

my\_list = [1, 2, 3]

my\_dict = {"a": 1, "b": 2}

my\_string = "hello"

Example:

X = 10

Y = 20

Print(x+y)##30

1. Bitwise operator: Bitwise operators are used to perform operations on the binary representations of numbers. They operate on the bits of the numbers, hence the name "bitwise".

1. AND (&)

2. OR (|)

3. XOR (^)

4. NOT (~)

5. Left Shift (<<)

6. Right Shift (>>)

1. AND (&): Performs a bit-by-bit AND operation.

Example: 5 & 3 = 1

2. OR (|): Performs a bit-by-bit OR operation.

Example: 5 | 3 = 7

3. XOR (^): Performs a bit-by-bit XOR operation.

Example: 5 ^ 3 = 6

4. NOT (~): Inverts all the bits.

Example: ~5 = -6

5. Left Shift (<<): Shifts the bits to the left.

Example: 5 << 2 = 20

6. Right Shift (>>): Shifts the bits to the right.

Example: 5 >> 2 = 1

Example:

X = 2.556

Y = 0.001

Print(X&Y)

1. Membership operator: Membership operators are used to test whether a value is a member of a sequence (such as a list, tuple, or string) or a dictionary. There are two membership operators:

- a = [1, 2, 3, 4, 5]

- 2 in a returns True

- 6 in a returns False

- 2 not in a returns False

- 6 not in a returns True

- s = "hello world"

- "hello" in s returns True

- "goodbye" in s returns False

- d = {"a": 1, "b": 2, "c": 3}

- "a" in d returns True

- "d" in d returns False

Example:

X = (3,4,5)

Y = 3,4 in Y

Print(y)## True

X = (3,4,5,6)

Y = 8 in X

Print(y)## False

1. Identity operators: Identity operators are used to compare the memory locations of two objects, i.e., whether they are the same object or not. There are two identity operators:

1. is: Returns True if both variables point to the same object in memory, and False otherwise.

2. is not: Returns True if both variables do not point to the same object in memory, and False otherwise.

For example:

a = [1, 2, 3]

b = [1, 2, 3]

a == b returns True (same value)

a is b returns False (different objects)