Assignment 1

Q.1 Explain all data types with examples.

Ans. Sure, here's an explanation of common data types typically found in programming languages, along with examples:

1. Integer (int):

Represents whole numbers without any decimal points.

Examples: 0, -10, 42, 123456.

2. Floating-point or Decimal (float, double):

Represents numbers with a decimal point.

Examples: 3.14, -0.001, 2.71828, 123.456.

3. Boolean (bool):

Represents a binary value that can be either True or False.

Examples: True, False.

4. Character (char):

Represents a single character, enclosed in single quotes (' ').

Examples: 'a', 'B', '1', '!'.

5. String (str):

Represents a sequence of characters, enclosed in double quotes (" ").

Examples: "Hello", "Python", "123", "!@#$".

6. Array:

Represents a collection of elements of the same type stored in contiguous memory locations.

Examples: In Python, lists can be considered analogous to arrays: [1, 2, 3, 4, 5].

7. Tuple:

Similar to an array, but its elements can be of different types and it is immutable.

Examples: (1, 'a', 3.14).

8. Dictionary (dict):

Represents a collection of key-value pairs where each key is unique.

Examples: {'name': 'John', 'age': 30, 'city': 'New York'}.

9. Set (set):

Represents an unordered collection of unique items.

Examples: {1, 2, 3, 4}, {'apple', 'banana', 'cherry'}.

10. NoneType (None):

Represents a null or undefined value.

Example: None.

Q.2 What is variable, how to create a variable?.

Ans. A variable in programming is a symbolic name or identifier that represents a value stored in the computer's memory. It allows programmers to store and manipulate data in their programs. Here's how you create a variable in a typical programming language (like Python, Java, C++, etc.):

1. Choose a Name: Decide on a meaningful name for your variable that reflects its purpose or the data it will hold. Variable names typically start with a letter or underscore (\_) and can contain letters, digits, and underscores.

2. Specify the Data Type: In some programming languages (like C, C++, Java), you need to specify the data type of the variable explicitly (e.g., integer, float, string). In dynamically-typed languages (like Python), the data type is inferred based on the value assigned to the variable.

3. Assign a Value: Use the assignment operator (=) to assign a value to the variable.

Here are example

// Creating variables in Java

String name = "Alice"; // a string variable

int age = 30; // an integer variable

double height = 5.8; // a double (floating-point) variable

boolean isStudent = true; // a boolean variable

// Variables can be updated with new values

age = 31;

In these example:

\* Name: name, age, height, is\_student, etc., are the names chosen for the variables.

\* Data Type: Variables are explicitly declared with their data types (string, int, double, boolean).

\* Assignment: Values ("Alice", 30, 5.8, True) are assigned to the variables using the = operator.

Q.3 Explain all operators with an example.

Ans. In Java, operators are symbols that perform operations on variables and values. Here’s an explanation of the common operators along with examples:

### Arithmetic Operators

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication, division, etc.

1. \*\*Addition (`+`)\*\*:

- Adds two operands.

```java

int a = 10;

int b = 20;

int c = a + b; // c is 30

```

2. \*\*Subtraction (`-`)\*\*:

- Subtracts the right operand from the left operand.

```java

int a = 20;

int b = 10;

int c = a - b; // c is 10

```

3. \*\*Multiplication (`\*`)\*\*:

- Multiplies two operands.

```java

int a = 5;

int b = 6;

int c = a \* b; // c is 30

```

4. \*\*Division (`/`)\*\*:

- Divides the left operand by the right operand.

```java

int a = 15;

int b = 4;

int c = a / b; // c is 3 (integer division, truncates the decimal part)

```

5. \*\*Modulus (`%`)\*\*:

- Returns the remainder of the division of the left operand by the right operand.

```java

int a = 15;

int b = 4;

int c = a % b; // c is 3 (remainder of 15 divided by 4)

```

6. \*\*Increment (`++`)\*\*:

- Increases the value of the operand by 1.

```java

int a = 5;

a++; // now a is 6

```

7. \*\*Decrement (`--`)\*\*:

- Decreases the value of the operand by 1.

```java

int a = 5;

a--; // now a is 4

```

### Relational Operators

Relational operators are used to compare values. They return a boolean result (`true` or `false`).

1. \*\*Equal to (`==`)\*\*:

- Checks if two operands are equal.

```java

int a = 10;

int b = 10;

boolean result = (a == b); // result is true

```

2. \*\*Not equal to (`!=`)\*\*:

- Checks if two operands are not equal.

```java

int a = 10;

int b = 20;

boolean result = (a != b); // result is true

```

3. \*\*Greater than (`>`)\*\*:

- Checks if the left operand is greater than the right operand.

```java

int a = 20;

int b = 10;

boolean result = (a > b); // result is true

```

4. \*\*Less than (`<`)\*\*:

- Checks if the left operand is less than the right operand.

```java

int a = 10;

int b = 20;

boolean result = (a < b); // result is true

```

5. \*\*Greater than or equal to (`>=`)\*\*:

- Checks if the left operand is greater than or equal to the right operand.

```java

int a = 20;

int b = 20;

boolean result = (a >= b); // result is true

```

6. \*\*Less than or equal to (`<=`)\*\*:

- Checks if the left operand is less than or equal to the right operand.

```java

int a = 10;

int b = 10;

boolean result = (a <= b); // result is true

```

### Logical Operators

Logical operators are used to combine conditional statements. They also return a boolean result.

1. \*\*Logical AND (`&&`)\*\*:

- Returns `true` if both operands are `true`.

```java

boolean a = true;

boolean b = false;

boolean result = (a && b); // result is false

```

2. \*\*Logical OR (`||`)\*\*:

- Returns `true` if either of the operands is `true`.

```java

boolean a = true;

boolean b = false;

boolean result = (a || b); // result is true

```

3. \*\*Logical NOT (`!`)\*\*:

- Inverts the boolean value of the operand.

```java

boolean a = true;

boolean result = !a; // result is false

```

### Bitwise Operators (for integers)

Bitwise operators perform operations on bits (binary digits) of integer operands.

1. \*\*Bitwise AND (`&`)\*\*:

- Performs bitwise AND operation.

```java

int a = 5; // 101 in binary

int b = 3; // 011 in binary

int c = a & b; // c is 1 (001 in binary)

```

2. \*\*Bitwise OR (`|`)\*\*:

- Performs bitwise OR operation.

```java

int a = 5; // 101 in binary

int b = 3; // 011 in binary

int c = a | b; // c is 7 (111 in binary)

```

3. \*\*Bitwise XOR (`^`)\*\*:

- Performs bitwise XOR (exclusive OR) operation.

```java

int a = 5; // 101 in binary

int b = 3; // 011 in binary

int c = a ^ b; // c is 6 (110 in binary)

```

4. \*\*Bitwise NOT (`~`)\*\*:

- Performs bitwise NOT operation (inverts all bits).

```java

int a = 5; // 101 in binary

int c = ~a; // c is -6 (in 32-bit signed representation, this is ~00000000000000000000000000000101 which is 11111111111111111111111111111010)

```

5. \*\*Left Shift (`<<`)\*\*:

- Shifts bits of the left operand left by the number of positions specified by the right operand.

```java

int a = 5; // 101 in binary

int c = a << 2; // c is 20 (10100 in binary, shifted left by 2 positions)

```

6. \*\*Right Shift (`>>`)\*\*:

- Shifts bits of the left operand right by the number of positions specified by the right operand.

```java

int a = 20; // 10100 in binary

int c = a >> 2; // c is 5 (101 in binary, shifted right by 2 positions)

```

### Assignment Operators

Assignment operators are used to assign values to variables.

1. \*\*Assignment (`=`)\*\*:

- Assigns the value on the right to the variable on the left.

```java

int a = 10;

```

2. \*\*Addition assignment (`+=`)\*\*:

- Adds the value on the right to the variable on the left and assigns the result to the variable on the left.

```java

int a = 5;

a += 3; // equivalent to a = a + 3; now a is 8

```

3. \*\*Subtraction assignment (`-=`)\*\*:

- Subtracts the value on the right from the variable on the left and assigns the result to the variable on the left.

```java

int a = 10;

a -= 2; // equivalent to a = a - 2; now a is 8

```

4. \*\*Multiplication assignment (`\*=`)\*\*:

- Multiplies the variable on the left by the value on the right and assigns the result to the variable on the left.

```java

int a = 5;

a \*= 2; // equivalent to a = a \* 2; now a is 10

```

5. \*\*Division assignment (`/=`)\*\*:

- Divides the variable on the left by the value on the right and assigns the result to the variable on the left.

```java

int a = 10;

a /= 2; // equivalent to a = a / 2; now a is 5

```

6. \*\*Modulus assignment (`%=`)\*\*:

- Computes the modulus of the variable on the left with the value on the right and assigns the result to the variable on the left.

```java

int a = 15;

a %= 4; // equivalent to a = a % 4; now a is 3

```

### Conditional Operator (Ternary Operator)

The conditional operator (`?:`) is a shorthand for if-else statements and is used to evaluate a boolean expression.

```java

int a = 10;

int b = 20;

int max = (a > b) ? a : b; // max is 20 (if a > b is true, max = a; otherwise, max = b)

```

### Type Comparison Operator (`instanceof`)

The `instanceof` operator is used to test whether an object is an

instance of a specific class or interface.

```java

Object obj = new Integer(10);

boolean result = obj instanceof Integer; // result is true

```

These examples cover the most commonly used operators in Java. Operators are fundamental for performing calculations, making decisions, and manipulating data in Java programs.