1. What are the differences between operators and values in the following?

**Ans**: Operators are used to perform operations on variables and values.

\*: This operator is used for multiplication of more than 1 value or variable.

'hello': This is a string data type.

-87.8: This is a float data type.

-: This operator is used for subtraction of more than 1 value or variable.

/: This operator is used for division of more than 1 value or variable.

+: This operator is used for additions of more than 1 value or variable.

6: This is int data type.

1. What is the difference between string and variable?

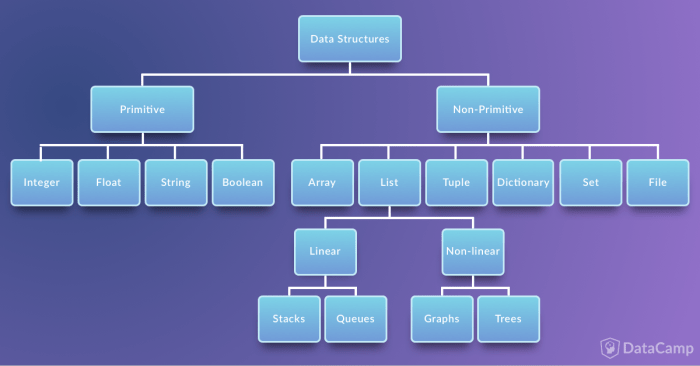
spam

'spam'

**Ans**: A Variable is a store of information, and a String is a type of information you would store in a variable. A String is usually words, enclosed with "" E.g., String x ="Test" x is the variable, and we declared it as a string, use the single = to assign the text to it.

1. Describe three different data forms.

**Ans**: Data structures are a way of organizing and storing data so that they can be accessed and worked with efficiently. They define the relationship between the data, and the operations that can be performed on the data. There are many various kinds of data structures defined that make it easier for the data scientists and the computer engineers, alike to concentrate on the main picture of solving larger problems rather than getting lost in the details of data description and access.



**1. Abstract Data Type and Data Structures:** Data structures help you to focus on the bigger picture rather than getting lost in the details. This is known as data abstraction. Now, data structures are actually an implementation of Abstract Data Types or ADT. This implementation requires a physical view of data using some collection of programming constructs and basic data types. Generally, data structures can be divided into two categories in computer science: primitive and non-primitive data structures. The former are the simplest forms of representing data, whereas the latter are more advanced: they contain the primitive data structures within more complex data structures for special purposes.

2. **Primitive Data Structures:** These are the most primitive or the basic data structures. They are the building blocks for data manipulation and contain pure, simple values of a data. Python has four primitive variable types:

* Integers: An integer represent numeric data, and more specifically, whole numbers from negative infinity to infinity, like 4, 5, or -1.
* Float: "Float" stands for 'floating point number'. You can use it for rational numbers, usually ending with a decimal figure, such as 1.11 or 3.14.
* Strings: Strings are collections of alphabets, words or other characters. In Python, you can create strings by enclosing a sequence of characters within a pair of single or double quotes. For example: ‘Take’, “Rest”, etc.
* Boolean: This built-in data type that can take up the values: True and False, which often makes them interchangeable with the integers 1 and 0. Booleans are useful in conditional and comparison expressions.

3.**Non-Primitive Data Structures:** Non-primitive types are the sophisticated members of the data structure family. They don't just store a value, but rather a collection of values in various formats. In the traditional computer science world, the non-primitive data structures are divided into:

* **Arrays:** Are a compact way of collecting basic data types, all the entries in an array must be of the same data type. arrays are supported by the arrays module and need to be imported before you start initializing and using them. The elements stored in an array are constrained in their data type. The data type is specified during the array creation and specified using a type of code, which is a single character like the I you see in the example below:

Import array as arr

a= arr.array (“I” , [1,2,3])

Type (a)

* **Lists:** Listsare used to store collection of heterogeneous items. These are mutable, which means that you can change their content without changing their identity. You can recognize lists by their square brackets [and] that hold elements, separated by a comma ,. Lists are built into Python: you do not need to invoke them separately.

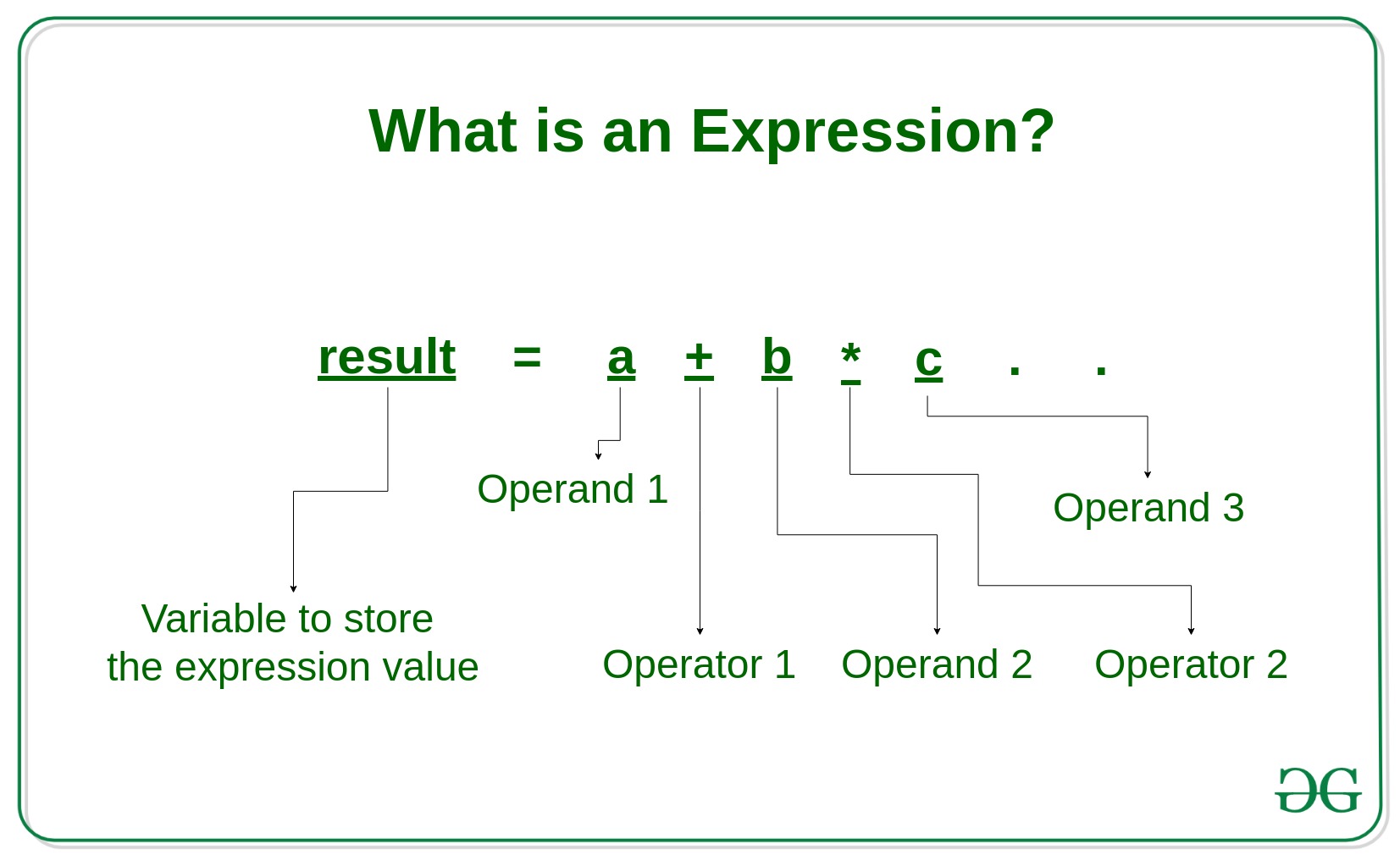
**X= [] #Empty list**

**Type (x)**

* **Files:** Files are traditionally a part of data structures. And although big data is commonplace in the data science industry, a programming language without the capability to store and retrieve previously stored information would hardly be useful. You still have to make use of the all the data sitting in files across databases and you will learn how to do this. The syntax to read and write files in Python is similar to other programming languages but a lot easier to handle. Here are some of the basic functions that will help you to work with files using Python:
* open () to open files in your system, the filename is the name of the file to be opened;
* read () to read entire files;
* readline () to read one line at a time;
* write () to write a string to a file, and return the number of characters written; And
* close () to close the file.

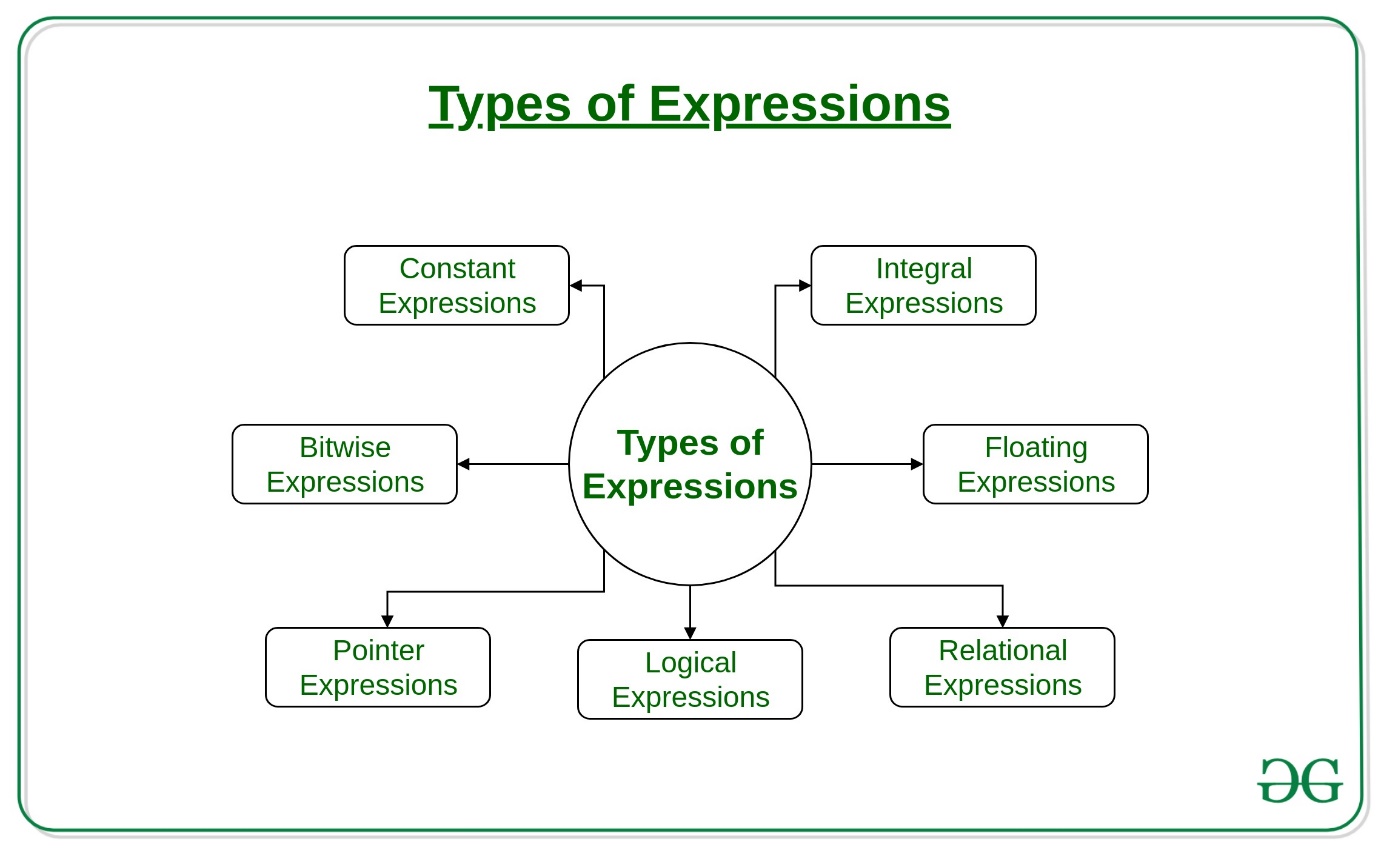
1. What makes up an expression? What are the functions of all expressions?

**Ans**: An expression is a combination of operators, constants, and variables. An expression may consist of one or more operands, and zero or more operators to produce a value.

[](https://media.geeksforgeeks.org/wp-content/uploads/20190801163131/What-is-an-Expression_-3.jpg)

**Types of Expressions:**

Expressions may be of the following types:

[](https://media.geeksforgeeks.org/wp-content/uploads/20190801154515/Types-of-Expressions.jpg)

* **Constant expressions**: Constant Expressions consists of only constant values. A constant value is one that does not change.  
  Examples:

5, 10 + 5 / 6.0, 'x’

* **Integral expressions**: Integral Expressions are those which produce integer results after implementing all the automatic and explicit type conversions.  
  Examples:

x, x \* y, x + int( 5.0)

where x and y are integer variables.

* **Floating expressions**: Float Expressions are which produce floating point results after implementing all the automatic and explicit type conversions.  
  Examples:

x + y, 10.75

where x and y are floating point variables.

* **Relational expressions**: Relational Expressions yield results of type bool which takes a value true or false. When arithmetic expressions are used on either side of a relational operator, they will be evaluated first and then the results compared. Relational expressions are also known as Boolean expressions.  
  Examples:

x <= y, x + y > 2

* **Logical expressions**: Logical Expressions combine two or more relational expressions and produces bool type results.  
  Examples:

x > y && x == 10, x == 10 || y == 5

* **Pointer expressions**: Pointer Expressions produce address values.  
  Examples:

&x, ptr, ptr++

where x is a variable and ptr is a pointer.

* **Bitwise expressions**: Bitwise Expressions are used to manipulate data at bit level. They are basically used for testing or shifting bits.  
  Examples:

x << 3

shifts three-bit position to left.

y >> 1

shifts one bit position to right.

Shift operators are often used for multiplication and division by powers of two.

1. In this chapter, assignment statements such as spam = 10 were added. What's the difference between a declaration and an expression?

**Ans**: **Function Declaration**: A Function Declaration (or a Function Statement) defines a function with the specified parameters without requiring a variable assignment. They exist on their own, i.e., they are standalone constructs and cannot be nested within a non-function block. A function is declared using the function keyword.

Syntax:

* function gfg(parameter1, parameter2) {
* //A set of statements
* }

**Function Expression**: A Function Expression works just like a function declaration or a function statement, the only difference is that a function name is NOT started in a function expression, that is, anonymous functions are created in function expressions. The function expressions run as soon as they are defined.

Syntax:

* var gfg = function (parameter1, parameter2) {
* //A set of statements

}

1. After running the following code, what does the variable bacon contain?

bacon = 22

bacon + 1

**Ans**: The bacon variable is set to 22. The bacon + 1 expression does not reassign the value in bacon (that would need an assignment statement: bacon = bacon + 1).

1. What should the values of the following two terms be?

'spam' + 'spamspam'

'spam' \* 3

**Ans**: Both expressions evaluate to the string 'spamspamspam'.

1. Why is it that eggs is a true variable name but 100 is not?

**Ans**: Variable names cannot begin with a number hence 100 is not a valid variable name.

1. Which of the following three functions may be used to convert a value to an integer, a floating-point number, or a string?

**Ans**: The int (), float(), and str() functions will evaluate to the integer, floating-point number, and string versions of the value passed to them.

1. What is the error caused by this expression? What would you do about it?

'I have eaten ' + 99 + ' burritos.'

**Ans**: The expression causes an error because 99 is an integer, and only strings can be concatenated to other strings with the + operator.

The correct way is I have eaten ' + str(99) + ' burritos.