Getting started with Python

Introduction to Python

- Program: A program is an ordered set of instructions, to be executed by a Computer, to carry out a specific task and language used to specify these set of instructions is called a Programming language.

  e.g.) Python, C++, C, Java etc.

- Why do we use programming language?
  - Computer uses machine language
  - We can't write and understand machine language
  - Compiler/interpreter is used to convert from

    to machine language

Introduction to Python

- Source Code: a program written in high-level language
- Python uses an interpreter to convert its instructions into machine language,

Compiler	Interpreter
translates the entire source code, as a whole, into the object code	processes the program statements one by one, first translating and then executing.
e.g.) C, C++, Java	e.g.) Python, PHP, Perl

Execution modes:

- There are two ways to use the Python interpreter:
  - Interactive mode
  - Script mode
- Interactive mode is convenient for testing a single line code for instant execution but in interactive mode, we can't save the statements for future and we have to retype the statements to run them again.
- In the script mode, we can write a Python program in a file, save it and then use the interpreter to execute it.

Python Glossary:

- Keywords: Keywords are reserved words which has a special meaning for Python interpreter. E.g.) True, for, while, else, elif etc.
- Identifiers: Identifiers are names used to identify a variable, function, or other entities in a program.
- Variable: Variable in Python refers to an object — an item or element that is stored in the memory.

Note:

Variables must always be assigned values before they are used in expressions as otherwise it will lead to an error in the program.

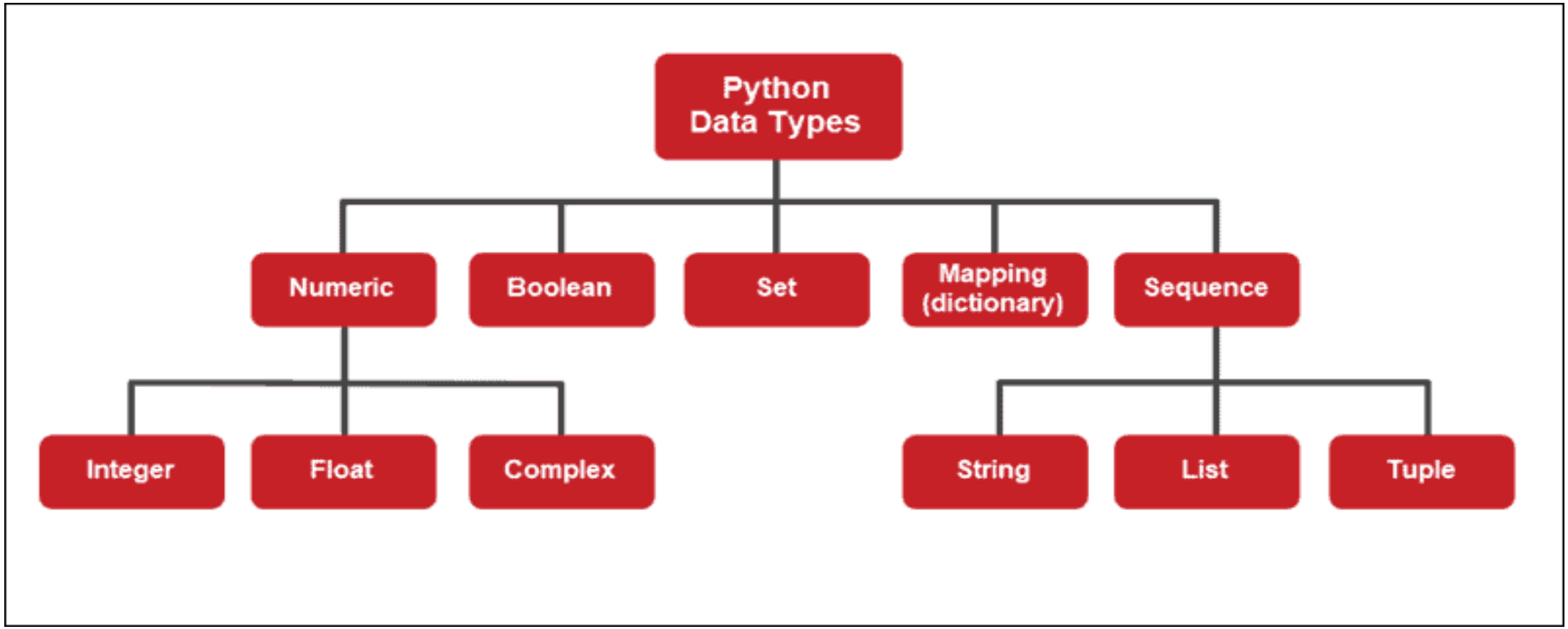
- Comments: single line comment, multi-line comment; not executed by the interpreter

Objects in Python

- Python treats every value or data item whether numeric, string, or other type as an object in the sense that it can be assigned to some variable or can be passed to a function as an argument.
- Every object in Python is assigned a unique identity (ID) which remains the same for the lifetime of that object.
- This ID is similar to the memory address of the object. The function id() returns the identity of an object.

```
In [12]: id(10)
Out[12]: 140711974904496

In [13]: id(30-20)
Out[13]: 140711974904496
```

Data types in Python



Data types in Python: Numeric

Type/class	Description	example
int	integer numbers	–12, –3, 0, 125, 2
float	real or floating point numbers	–2.04, 4.0, 14.23
complex	complex numbers	3 + 4j, 2 – 2j

- Boolean data type (bool) is a subtype of integer, consisting of 2 constants:
  - True
  - False

Data types in Python: Sequence

- A Python sequence is an ordered collection of items, where each item is indexed by an integer.
  - Strings
  - Lists
  - tuples

Strings	Lists	Tuple
Group of characters	sequence of items separated by commas	Like list, tuple is a sequence of items, separated by commas
Enclosed in either single quotes (‘ ’) or double quotes	items are enclosed in square brackets [ ]	items are enclosed in parenthesis ( )
Immutable data type	Mutable data type	Immutable data type
e.g.) “de4dr”, ‘fre67’	[5, 3.4, "New Delhi", "20C", 45]	(10, 20, "Apple", 3.4, 'a')

Data types in Python: Set

- Set is an unordered collection of items separated by commas, and
- the items are enclosed in curly brackets { }

  e.g.) {10,20,3.14,"New Delhi"}

Note:

A set is similar to list, except that

- it cannot have duplicate entries
- It is immutable, and
- It is unordered

Data types in Python: Mapping

- Mapping is an unordered data type in Python
- Currently, there is only one standard mapping data type in Python called dictionary

  e.g.) {'Fruit': 'Apple', 'Climate': 'Cold', 'Price(kg)': 120}

Data types in Python: Dictionary

- Dictionary in Python holds data items in key-value pairs, where every key is separated from its value using a colon (:) sign

  e.g.) dict1= {'Fruit‘ : 'Apple', 'Climate‘ : 'Cold', 'Price(kg)‘ : 120}

Note:

key is usually strings but it can be any immutable data type like number, string and tuple, and their values can be of any data type

- In order to access any value in the dictionary, we have to specify its key in square brackets [ ]

  e.g.) print(dict1['Price(kg)'])

Mutable and Immutable data types

- Mutable: Variables whose values can be changed after they are created and assigned are called mutable

  E.g.) Lists, dictionary, etc.

- Immutable: Variables whose values cannot be changed after they are created and assigned are called immutable.

  E.g.) Numbers, strings, tuples etc.

What happens when we try to update a variable?

is value changed or the location changed

When to use which Data type?

- List: when we require collection of data that requires frequent modifications

  e.g.) entry register in a hotel

- Tuple: when data doesn't require any change

  E.g.) name of week's days

- Sets: when we need uniqueness of data, and avoid duplicity

  E.g.) name of month's name,

- Dictionary: when we need lookup of data using key, or we need a logical association between the *key : value* pair

  E.g.) Mobile phone book, dictionary

Operators in Python

- An operator is used to perform specific mathematical or logical operation on values (aka operands)

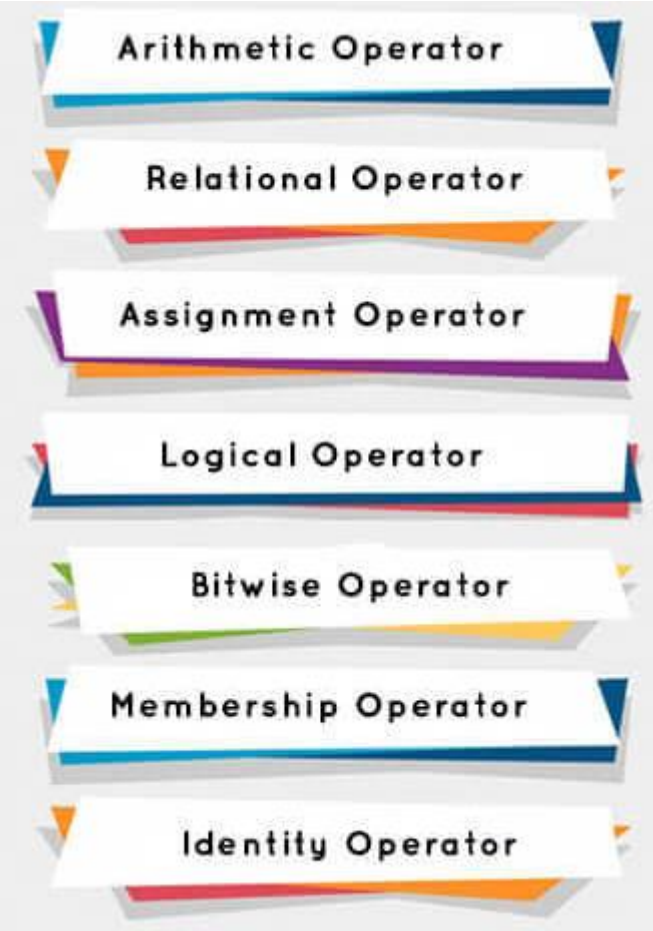
  E.g.) 10 + *num*

  here, 10 and num are operands,

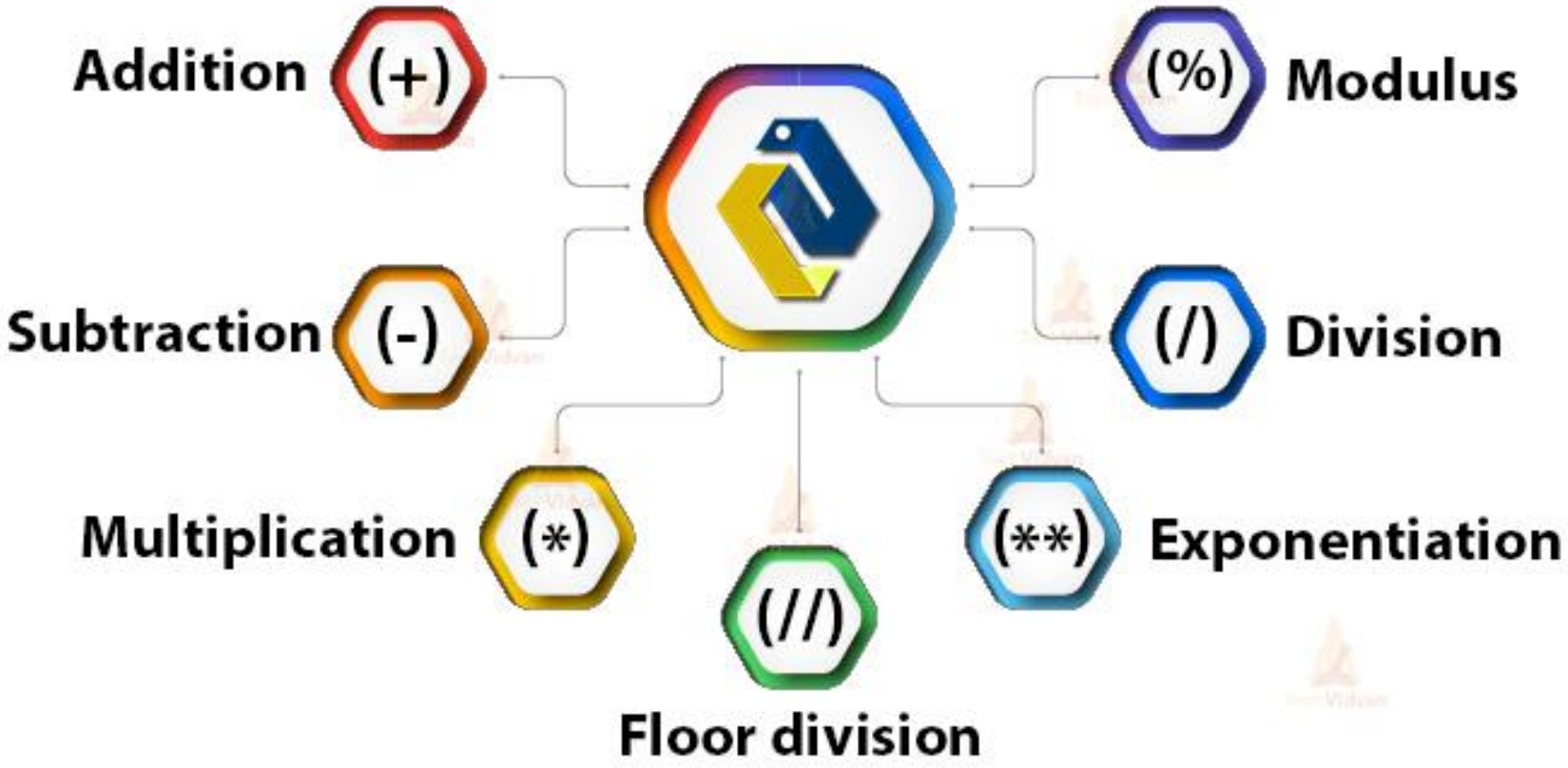
  + (plus) sign is an operator, and

  10 + num is an expression

- In Python, we have 7 types of Operators:



1. Arithmetic Operators



1. Arithmetic Operators

Operator	Operation	Description	Example
+	Addition	can also be used to concatenate two strings on either side of the operator	a = “Hello”, b = “World” a + b = “HelloWorld”
*	Multiplication	Repeats the item on left of the operator - if first operand is a string and - second operand is an integer value	a = “hi”, b = 3 a * b = “hihihi”
/	Division	Divides the operands and returns the result in float	a =10, b = 12 a / b = 0.8333333333333334
%	Modulus	Gives remainder	a =10, b = 12 a % b = 10
//	Floor Division	performs integer division	a =10, b = 12 a // b = 0
**	Exponent	Gives power i.e., pow(a,b)	a = 2, b = 4 a**b = 16

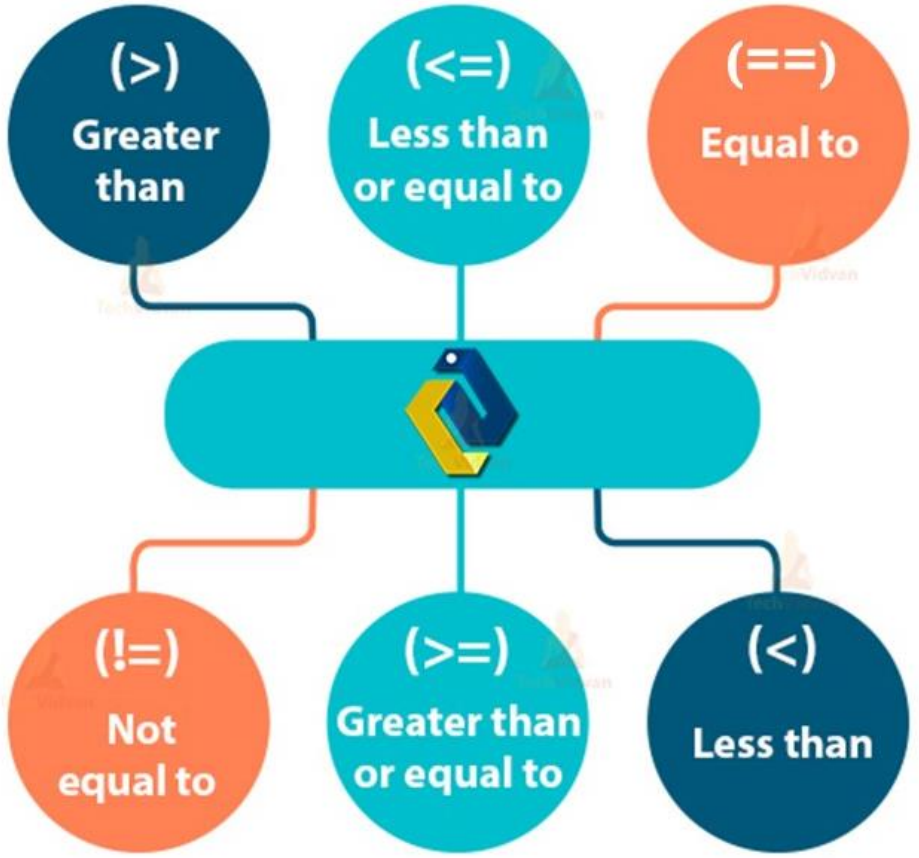
2. Relational Operators

- Also known as comparison operators, relational operators compares 2 operands to give output

E.g.)

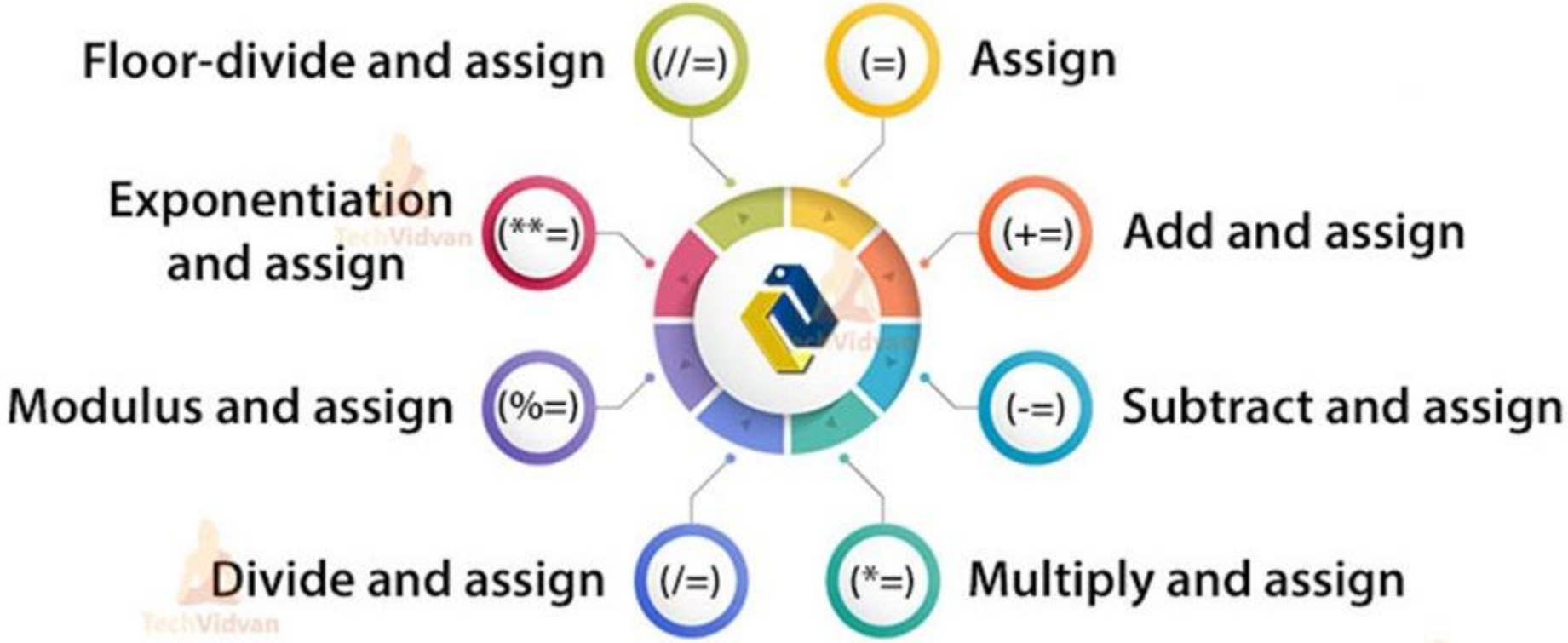
```
In [15]: 45<78
Out[15]: True

In [16]: "asvd">"cxvg"
Out[16]: False

In [17]: {1,2,3} > {2,3,4}
Out[17]: False
```

- We have 6 relational operators



3. Assignment Operators

- Assignment operator assigns or changes the value of the variable on its left.



4. Logical Operators

- There are three logical operators supported by Python:
  1. and
  2. or
  3. not

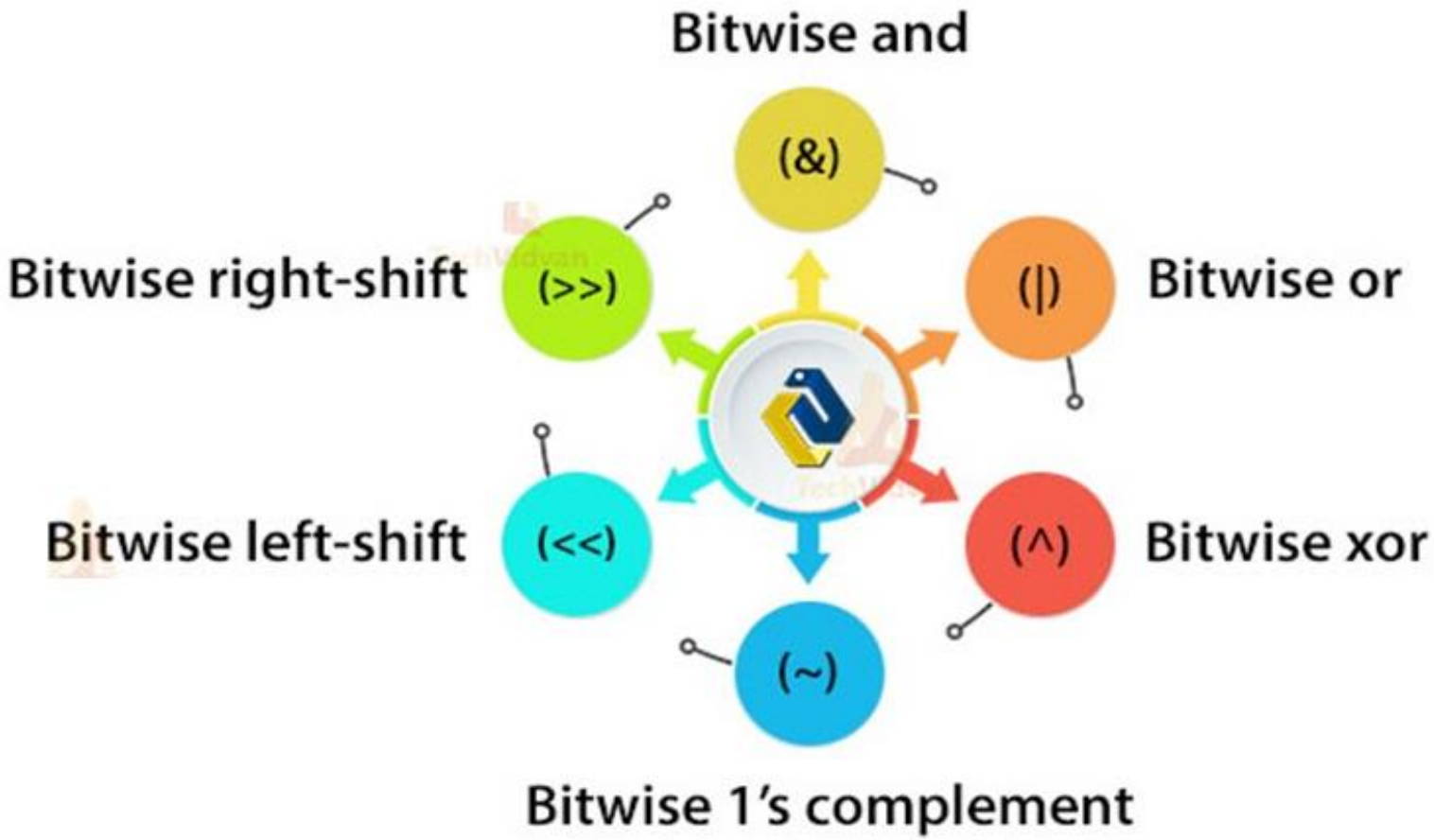
Note:

Every value is *logically* either True or False. By default, all values are True except None, False, 0 (zero), empty collections "", (), [], {}, and few other special values

4. Logical Operators

Operator	Operation	Description	Example
and	Logical AND	If both the operands are True, then condition becomes True	a = 10, b =20 bool(a and b) = True
or	Logical OR	If any of the two operands are True, then condition becomes True	a = 10, b =20 bool(a or b) = True
not	Logical NOT	Used to reverse the logical state of its operand	a = 10 bool(not a) = False

5. Bitwise Operator



6. Membership Operator

- Membership operators are used to check if a value is a member of the given sequence or not.

Operator	Description	Example

in	Returns True if the variable/value is found in the specified sequence and False otherwise	>>> a = [1,2,3]
		>>> 2 in a
		True
		>>> '1' in a
		False

not in	Returns True if the variable/value is not found in the specified sequence and False otherwise	>>> a = [1,2,3]
		>>> 10 not in a
		True
		>>> 1 not in a
		False

7. Identity Operator

- Identity operators are used to determine whether the value of a variable is of a certain type or not.

E.g.)

```
In [26]: a = 5

In [27]: type(a) is int
Out[27]: True
```

- Identity operators can also be used to determine whether two variables are referring to the same object or not.

e.g.)

```
In [29]: a = 10

In [30]: b = 30-20

In [31]: a is b
Out[31]: True
```

Expressions in Python

- An expression is defined as a combination of constants, variables, and operators. An expression always evaluates to a value.

  E.g.) 23/3 -5 * 7(14 -2), "Global" + "Citizen“

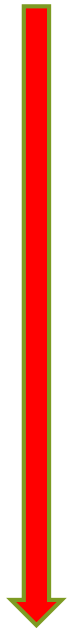
Note:

A value or a standalone variable is also considered as an expression but a standalone operator is not an expression.

- 120 is an expression
- + is not an expression
- 120 + num is an expression

Precedence of Operators

highest



lowest

Order of Precedence	Operators	Description
1	**	Exponentiation
2	~, +, -	Unary Operator
3	* ,/, %, //	Multiply, divide, modulo and floor division
4	+, -	Addition and subtraction
5	<= , < , > , >=, == , !=	Relational and Comparison operators
6	=, %=, /=, //=, -=, +=,*=, **=	Assignment operators
7	is, is not	Identity operators
8	in, not in	Membership operators
9	not > and > or	Logical operators

Precedence of Operators: Key points

- Parenthesis can be used to override the precedence of operators. The expression within () is evaluated first.

  e.g.) (2 + 3) * 4 = 6 * 4 = 24

- For operators with equal precedence, the expression is evaluated from left to right.

  e.g.) 20 - 30 + 40 = -10 + 40 = 30

Q.) evaluate 12 / 5 + (8.5 + 2.3)

Input and Output in Python

- In general, a program requires some inputs from user, so that it can process this data and produce/display the required output.

- In Python,
  - we have function input() for taking the user input, and
  - Function print()to print out the data to the screen

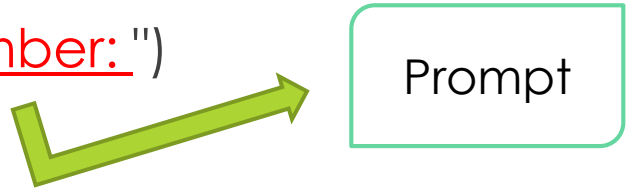
Input function: input()

- input() function accepts all user input as string. The user may enter a number or a string but the input() function treats them as strings only.
- The syntax for input() is:

  input ([Prompt])

  where Prompt is the string we may like to display on the screen prior to taking the input, and it is optional

  e.g.) num = input("Enter your number: ")



Program-1: Input a welcome message and display it.

- Code:

'''

Program - 1: Input a welcome message and display it.

@author: Himanshu Mudgal

'''

#take the input

message = input("Enter a welcome message: ")

print(message)

- Output:

>>> Enter a welcome message: Hi, How are you?

Hi, How are you?

```
"""
Program - 1: Input a welcome message and display it.
@author: Himanshu Mudgal
"""
#take the input
message = input("Enter a welcome message: ")
print(message)
```

Type conversion

- There are two ways to do type-conversion:
  - Explicit Conversion: when programmer forced it
  - Implicit Conversion: when interpreter does it automatically

e.g.) int(x): converts x to an integer

str(x): converts x to a string

Note:

Python interpreter does the type casting (Implicit type casting) by converting data into a wider-sized data type without any loss of information

Debugging

- Errors occurring in programs can be categorised as:
  - i) Syntax errors: program will not run
  - ii) Logical errors: program will not give desired output
  - iii) Runtime errors: program terminates abruptly

e.g.)

```
(10 + 2
10 + 20 / 2
10 / 0
```

Assignment -2:

- Read chapter number – 5 of class 11th, especially,
  - 5.7.6: Mutable and immutable data types; what happens when we try to update a variable?
  - operators in Python
  - Table 5.9: Precedence of operators in Python
- List the keywords of Python in your notebook, (table 5.1, page 90-91)
- Note down the rules for naming an identifier in your notebook (4 rules)
- Read the summary

Programming Assignment -1:

1. Write a program to find the average of three numbers.

2. Write a program that accepts the value of radius of a sphere and compute its
   - surface area, and
   - volume

3. Write a program to repeat the string ‘‘GOOD MORNING” n times. Here ‘n’ is an integer entered by the user.