MODULE 1

Introduction to python, features, IDLE, python interpreter, Writing and executing python scripts, comments, identifiers, keywords, variables, data type, operators, operator precedence and associativity, statements, expressions, user inputs, type function, eval function, print function

- Python is an open source, object-oriented, high-level powerful programming language.
- Developed by Guido van Rossum in 1991 at National Research Institute for Mathematics and Computer Science,Netherlands,It is presently owned by the Python Software Foundation.
- Named after Monty Python
- Python runs on many Unix variants, on the Mac, and on Windows 2000 and later.
- Latest version of python is python 3.11.5, documentation released on 24 August 2023.

FEATURES

- Python is an interpreted, interactive, directly executed language with precompiled code.
- It is a loosely typed object oriented programming language with few keywords (reserved words), simple English like structure and is easy to learn.
- It is free, open source and portable language having a large repository of libraries.
- Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Extendable − you can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- Databases − Python provides interfaces to all major commercial databases.
- GUI Programming − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- Scalable − Python provides a better structure and support for large programs than shell scripting.
- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++,Java etc.

PYTHON - ENVIRONMENT SETUP

Python is available on a wide variety of platforms including Linux and Mac OS X.

Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python https://www.python.org/

You can download Python documentation from https://www.python.org/doc/. The documentation is available in HTML, PDF, and PostScript formats.

Installing Python

Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for your platform and install Python.

If the binary code for your platform is not available, you need a C compiler to compile the source code manually. Compiling the source code offers more flexibility in terms of choice of features that you require in your installation.

Installing Python on various platforms −

Unix and Linux Installation

- Here are the simple steps to install Python on Unix/Linux machine.
- Open a Web browser and go to https://www.python.org/downloads/.
- Follow the link to download zipped source code available for Unix/Linux.
- Download and extract files.
- Editing the Modules/Setup file if you want to customize some options.
- run ./configure script and install.
- This installs Python at standard location /usr/local/bin and its libraries at /usr/local/lib/python312 where 312 is the version of Python.

Windows Installation

- Here are the steps to install Python on Windows machine.
- Open a Web browser and go to https://www.python.org/downloads/.
- Follow the link for the Windows installer python-3.1.2.msi file where 3.1.2 is the version you need to install.
- To use this installer python-3.1.2.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

Setting path at Windows

To add the Python directory to the path for a particular session in Windows −At the command prompt − type path %path%;C:\Python and press Enter.

Note − C:\Python is the path of the Python director

Running Python

There are three different ways to start Python −

1. **Interactive Interpreter**

   You can start Python from Unix, DOS, or any other system that provides you a command-line interpreter or shell window.Enter python the command line.Start coding right away in the interactive interpreter.

   $python # Unix/Linux

   or

   python% # Unix/Linux

   or

   C:> python # Windows/DOS

2. **Script from the Command-line**

   A Python script can be executed at command line by invoking the interpreter on your application, as in the following −

   $python script.py # Unix/Linux

   or

   python% script.py # Unix/Linux

   or

   C: >python script.py # Windows/DOS

   Note − Be sure the file permission mode allows execution.

3. **IDLE**

   To write and run python programs interactively, we can either use the command line window or the IDLE. Integrated Development Environment or Integrated Development and Learning Environment. It is an integrated development environment (IDE) for Python. The most important feature of IDLE is that it is a program that allows the user to edit, run, browse and debug a python program from a single interface.

Python Shell

When you first start python IDLE by clicking on its icon created on the desktop or menu item on the start menu->Apps by name->IDLE,it always starts up in the shell.

Python shell is an interactive window where you can type in the python code and see the output in the same window.it is an interface between python commands and the OS.

Python IDLE Comprises Python Shell(Interactive mode) and Python Editor(Script mode).

- The command prompt (>>>) followed by a blinking cursor gets displayed which indicates that IDLE is now ready to take Python commands from the user.
- The three greater than signs (>>>) are called the prompt or python command prompt.
- When commands are entered directly in IDLE from the keyboard, the Interpreter IDLE is said to be in Interactive mode.

Script Mode:

In this mode source code is stored in a file with the .py extension and use the interpreter to execute the contents of the file. To execute the script by the interpreter, you have to tell the interpreter the name of the file.

Example: if you have a file name demo.py, to run the script you have to follow the following steps:

Step-1: Open the text editor i.e. Notepad

Step-2: Write the python code and save the file with .py file extension. (Default directory is C:\Python33/Demo.py)

Step-3: Open IDLE (Python GUI) python shell

Step-4: Click on file menu and select the open option

Step-5: Select the existing python file

Step-6: Now a window of python file will be opened

Step-7: Click on Run menu and the option Run Module.

Step-8: Output will be displayed on python shell window

PYTHON COMMENTS

Comments in Python are the lines in the code that are ignored by the interpreter during the execution of the program. Comments enhance the readability of the code and help the programmers to understand the code.

Comments starts with a #, and Python will ignore them

```
#This is a comment
print("Hello, World!")
```

Comments can be placed at the end of a line, and Python will ignore the rest of the line:

```
print("Hello, World!") #This is a comment
```

To add a multiline comment you could insert# for each line:

```
#This is a comment
#written in
#more than just one line
```

Since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it:

```
"""
This is a comment
written in
more than just one line
"""
print("Hello, World!")
```

PYTHON CHARACTER SET

Character set is the set of valid characters that a language can recognize. A character represents any letter, digit or any other symbol. Python has the following character sets:

- **Letters** – A to Z, a to z
- **Digits** – 0 to 9
- **Special Symbols** - + - * / () {}[]etc.
- **Whitespaces** – Blank Space, tab, carriage return, newline, form feed
- **Other characters** – Python can process all ASCII and Unicode characters as part of data or literals.

TOKENS

A token is the smallest individual unit in a python program. All statements and instructions in a program are built with tokens. Python breaks each logical line into a sequence of elementary lexical components

Known as Tokens.

Python has the following tokens:

(i) Keyword

(ii) Identifiers

(iii) Literals

(iv) Operators

(v) Punctuator

KEYWORDS

Keywords are special words used by Python interpreter to recognize the structure of program. As these words have specific meaning for interpreter, they cannot be used for any other purpose. E.g. int, float,True,elif,else,break,as etc

For checking/displaying the list of keywords available in python use

>>>import keyword

>>>print(keyword.kwlist)

IDENTIFIERS

An Identifier is a name used to identify a variable, function, class, module or object.

E.g.: Sum, total_marks, regno, num1, Column31, _Total

Rules for Identifier:

- It consist of letters and digits in any order except that the first character must be a letter letter or an underscore (_) but no a digit.
- The underscore (_) counts as a character.
- Spaces are not allowed.
- Special Characters are not allowed. Such as @,$ etc.
- An identifier must not be a keyword of Python.
- Variable names should be meaningful which easily depicts the logic.
- Uppercase and lowercase are treated differently.

LITERALS OR VALUES

Literals are the fixed values or data items used in a source code. They are also known as constants.

- Python supports different types of literals such as:
- String literals:eg a=”hello”
- Numeric(integer)literals:eg p=2,a=100
- Floating point literals:eg salary=15000.00
- Boolean literals:eg value1=True value2=False
- Special literals:eg None

Boolean literals

Used to represent one of the two boolean values, that is True and False.

```
>>>bool_1=(6>10)
>>>print (bool_1)
    False
```

Special literals-· None

It is used to indicate something that has not been created yet or to signify the absence of values in a situation. Python doesn't display anything when we give a command to display the value of a variable containing value as None.

For example:

```
>>>value1=20
>>>value2=None
>>>value1
```

PYTHON DATA TYPES

Python Data types are used to define the type of a variable. It defines what type of data we are going to store in a variable. The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters.

type() function:

type() function is a built-in function in python that allows the programmer to evaluate the type of data used in the program for every variable.eg:

```
a = 10
b = 10.5
c = True
d = 1 + 5j
print(type(a))
print(type(b))
print(type(c))
print(type(d))
```

```
<class 'int'>
<class 'float'>
<class 'bool'>
<class 'complex'>
```

Python has the following data types built-in by default, in these categories:

Data Types	Classes	Description
Numeric	int, float, complex	holds numeric values
String	str	holds sequence of characters
Sequence	list, tuple, range	holds collection of items
Mapping	dict	holds data in key-value pair form
Boolean	bool	holds either True or False
Set	set, frozeenset	hold collection of unique items

Numeric Data Type in Python

The numeric data type in Python represents the data that has a numeric value. A numeric value can be an integer, a floating number, or even a complex number. These values are defined as int, float, and complex classes in Python.

Integers – This value is represented by int class. It contains positive or negative whole numbers (without fractions or decimals). In Python, there is no limit to how long an integer value can be.

Integers can be binary, octal, and hexadecimal values.

```
>>>b = 0b11011000 # binary
>>>print(b)
216
>>>o = 0o12 # octal
>>>print(o)
10
>>>h = 0x12 # hexadecimal
>>>print(h)
18
```

int() function converts a string or float to int.

```
>>>x = int('100')
>>>print(x)
100
```

Float – In Python, floating point numbers (float) are positive and negative real numbers with a fractional part denoted by the decimal symbol or the scientific notation E or e, e.g. 1234.56, 3.142, -1.55, 0.23.

```
>>>f = 1.2
>>>print(f)
1.2
>>>print(type(f))
<class 'float'>
>>>f = 1e3
>>>print(f)
1000.0
```

float() function to convert string, int to float.

```
>>>f=float('5.5')
>>>print(f)      #output: 5.5
```

Complex Numbers – Complex number is represented by a complex class. It is specified as (real part) + (imaginary part)j. For example – 2+3j

You must use j or J as imaginary component. Using other character will throw syntax error

```
>>>a = 5
>>>print("Type of a: ", type(a))
<class 'int'>
>>>b = 5.0
>>>print("\nType of b: ", type(b))
<class 'float'>
>>>c = 2 + 4j
>>>print("\nType of c: ", type(c))
<class 'complex'>
```

Sequence Data Type in Python

The sequence Data Type in Python is the ordered collection of similar or different data types. Sequences allow storing of multiple values in an organized and efficient fashion.

- Python String
- Python List
- Python Tuple

Strings in Python are arrays of bytes representing Unicode characters. A string is a collection of one or more characters put in a single quote, double-quote, or triple-quote. In python there is no character data type, a character is a string of length one. It is represented by **str** class.

Creating String

Strings in Python can be created using single quotes or double quotes or even triple quotes.

```
String1 = 'Welcome to the Geeks World'
print("String with the use of Single Quotes: ")
print(String1)

String1 = "I'm a Geek"
print("\nString with the use of Double Quotes: ")
print(String1)

String1 = '''I'm a Geek and I live in a world of "Geeks"'''
```

```
print("\nString with the use of Triple Quotes: ")
print(String1)
```

Escape Sequence

Python allows having certain non-graphic characters in string values. Such characters cannot be typed directly from keyboard.e.g. backspace, tabs, carriage return etc.

These can be represented by escape sequence. An escape sequence is represented by backslash followed by one or more characters. Following are the list of escape sequence:

\" - Double quotes

\' - Single quotes

\\ - Backslash

\a - Bell

\b - Backspace

\n - New line

\r - Carriage return

\t - Horizontal tab

\v - Vertical tab

PUNCTUATORS/DELIMITERS

These are the symbols that used in Python to organize the structures, statements, and expressions. Some of the Punctuators are: [ ] { } ( ) @ -= += *= //= **== = , etc.

OPERATORS

In computer programming languages operators are special symbols which represent computations, conditional matching etc. The variables on which operation is applied are called operands. Operators can be unary or binary. Unary operators are the ones acting on a single operand like complement operator, etc. While binary operators need two operands to operate. Python includes the following categories of operators:

1. **Arithmetic Operators**
2. **Assignment Operators**
3. **Comparison Operators**
4. **Logical Operators**
5. **Identity Operators**
6. **Membership Test Operators**
7. **Bitwise Operators**

Arithmetic Operators

Assume variable **a=10** and variable **b =20**, then

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 30
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -10
* Multiplication	Multiplies values on either side of the operator	a * b = 200
/ Division	Divides left hand operand by right hand operand	b / a = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 0
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity) −	9//2 = 4 9.0//2.0 = 4.0 -11//3 = -4, -11.0//3 = -4.0

Python Comparison Operators

These operators compare the values on either sides of them and decide the relation among them. They are also called Relational operators.Assume variable **a =10** and **b =20**, then

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	(a != b) is true.
<>	If values of two operands are not equal, then condition becomes true.	(a <> b) is true. This is similar to != operator.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.

<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

Python Assignment Operators

Assume variable a holds 10 and variable b holds 20, then

Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / a
%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**= Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//= Floor Division	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

Python Bitwise Operators

Bitwise operator works on bits and performs bit by bit operation. Assume if a = 60; and b = 13; Now in the binary format their values will be 0011 1100 and 0000 1101 respectively. Following table lists out the bitwise operators supported by Python language with an example each in those, we use the above two variables (a and b) as operands −

a = 0011 1100

b = 0000 1101

---

a&b = 0000 1100

a|b = 0011 1101

a^b = 0011 0001

~a = 1100 0011

Operator	Description	Example
& Binary AND	Operator copies a bit to the result if it exists in both operands	(a & b)=12(means 0000 1100)
| Binary OR	It copies a bit if it exists in either operand.	(a | b) = 61 (means 0011 1101)
^ Binary XOR	It copies the bit if it is set in one operand but not both.	(a ^ b) = 49 (means 0011 0001)
~ Binary Ones Complement	It is unary and has the effect of 'flipping' bits.	(~a) = -61 (means 1100 0011 in 2's complement form due to a signed binary number.
<< Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.	a << 2 = 240 (means 1111 0000)
>> Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	a >> 2 = 15 (means 0000 1111)

Python Logical Operators

There are following logical operators supported by Python language. Assume variable a holds 10 and variable b holds 20 then

Operator	Description	Example
and Logical AND	If both the operands are true then condition becomes true.	(a and b) is true.
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b) is true.
not Logical NOT	Used to reverse the logical state of its operand.	Not(a and b) is false.

Python Membership Operators

Python's membership operators test for membership in a sequence, such as strings, lists, or tuples. There are two membership operators

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

Python Identity Operators

Identity operators compare the memory locations of two objects. There are two Identity operators explained below −

Operator	Description	Example
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here **is** results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here **is not** results in 1 if id(x) is not equal to id(y).

OPERATOR PRECEDENCE AND ASSOCIATIVITY OF OPERATORS IN PYTHON

The combination of values, variables, operators, and function calls is termed as an expression. The Python interpreter can evaluate a valid expression.For example:

```
>>> 5 - 7
-2
```

Here 5 - 7 is an expression. There can be more than one operator in an expression.To evaluate these types of expressions there is a rule of precedence in Python. It guides the order in which these operations are carried out.

Operators	Meaning
()	Parentheses
**	Exponent
+x, -x, ~x	Unary plus, Unary minus, Bitwise NOT
*, /, //, %	Multiplication, Division, Floor division, Modulus
+, -	Addition, Subtraction
<<, >>	Bitwise shift operators
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
==, !=, >, >=, <, <=, is, is not, in, not in	Comparisons, Identity, Membership operators
not	Logical NOT
and	Logical AND
or	Logical OR

When two operators have the same precedence, associativity helps to determine the order of operations.

Associativity is the order in which an expression is evaluated that has multiple operators of the same precedence. **Almost all the operators have left-to-right associativity**.

Note: In Python, the exponent operation ** possesses the right to left associativity.

Evaluate 12+3*4-6/2

=12+12-6/2

=12+12-3.0

=21.0

Non Associative Operators: Some operators like assignment operators and comparison operators do not have associativity in Python. Thus, we have separate rules for sequences of this kind of operator and we cannot express them as associativity.

For instance, x < y < z neither means (x < y) < z nor x < (y < z). x < y < z is equivalent to x < y and y < z, and is evaluated from left-to-right.

PYTHON STATEMENT AND EXPRESSION

- A **statement** is an instruction that the Python interpreter can execute. We have seen two kinds of statements: **print and assignment.**
- When you type a statement on the command line, Python executes it and displays the result, if there is one. The result of a print statement is a value.
- Assignment statements don't produce a result. some other kinds of statements in Python are – if statement, else statement, while statement, for statement, import statement, etc.
- An **Expression** is a sequence or combination of values, variables, operators and function calls that always produces or returns a result value.
- An Expression always evaluates (calculate) to itself.

Example:

x = 5, y = 3, z = x + y

In the above example **x, y and z are variables, 5 and 3 are values, = and + are operators.**

So, the first combination **x = 5 is an expression**, the second combination **y = 3 is an another expression** and at last, **z = x + y is also an expression**.

Evaluation of an expression: It simply means that, to calculate or to solve the value of something. Such as in the above example: z = x + y, which is equivalent to z = 5 + 3. So, here we are calculating the value of z. and, hence, z = 8.

USER INPUT

Python provides two built-in methods to read the data from the keyboard. These methods are

- **input(prompt)**
- **raw_input(prompt**)

input():this function is used to get data from the user while working with script mode.it enables us to accept an input string from user without evaluation its value.The Python interpreter automatically identifies the whether a user input a string, a number, or a list.

Example -

```
name = input("Enter your name: ")
print(name)
```

Output:

```
Enter your name: Devansh
Devansh
```

The Python interpreter will not execute further lines until the user enters the input.

Example - 2

```
name = input("Enter your name: ")  # String Input
age = int(input("Enter your age: ")) # Integer Input
marks = float(input("Enter your marks: ")) # Float Input
print("The name is:", name)
print("The age is:", age)
print("The marks is:", marks)
```

Output:

```
Enter your name: Johnson
Enter your age: 21
Enter your marks: 89
The name is: Johnson
The age is 21
The marks is: 89.0
```

By default, the input() function takes input as a string so if we need to enter the integer or float type input then the input() function must be type casted.

```
age = int(input("Enter your age: ")) # Integer Input
marks = float(input("Enter your marks: ")) # Float Input
```

raw_input() - The raw_input function is used in Python's older version like Python 2.x. It takes the input from the keyboard and return as a string. The Python 2.x doesn't use much in the industry.eg:

```
name = raw_input("Enter your name : ")
print name
```

Output:

Enter your name: Peter

Peter

eval()

eval() is a built-in- function used in python, eval() function parses the expression argument and evaluates it as a python expression. The eval() function evaluates the “String” like a python expression and returns the result as an integer.

Syntax

```
eval(expression, [globals[, locals]])
```

The eval() function takes three parameters:

expression - the string parsed and evaluated as a Python expression

globals (optional) - a dictionary

locals (optional)- a mapping object. Dictionary is the standard and commonly used mapping type in Python.

The eval() method returns the result evaluated from the expression.

```
x = 1
print(eval('x + 1'))
```

print() function

- The print() function prints the specified message to the screen, or other standard output device.
- The message can be a string, or any other object, the object will be converted into a string before written to the screen.

Syntax

print*(object(s)*, sep=*separator*, end=*end*, file=*file*, flush=*flush*)

Parameter	**Description**
object(s)	Any object, and as many as you like. Will be converted to string before printed
sep='*separator*'	Optional. Specify how to separate the objects, if there is more than one. Default is ' '
end='*end*'	Optional. Specify what to print at the end. Default is '\n' (line feed)
file	Optional. An object with a write method. Default is sys.stdout
flush	Optional. A Boolean, specifying if the output is flushed (True) or buffered (False). Default is False

```
print("Hello", "how are you?")        # Hello how are you?
print("Hello", "how are you?", sep="---",end='\n')       # Hello---how are you?
```