Chapter 1 Fundamentals of Python

1.1 Introduction to Python

- ✔ Python stands out as a widely employed high-level, general-purpose programming language.
- ✔ It operates as an interpreted, object-oriented language.
- ✔ Python is not only freely available but also open-source, ensuring easy accessibility for users.
- ✔ A primary focus of Python is on code reusability, readability, and the use of whitespace.
- ✔ It boasts characteristics like high-level built-in data structures, dynamic typing, and binding, all of which render it appealing for swift application development.
- ✔ Originally conceived with an emphasis on code readability, Python's syntax enables programmers to express concepts succinctly, resulting in fewer lines of code.
- ✔ Python serves as a programming language that facilitates rapid work and enhances system integration efficiency.

❖ History of Python

- ✔ Guido van Rossum was the original designer of Python in 1991, and its development has been overseen by the Python Software Foundation.
- ✔ Python has two significant versions: Python 2 and Python 3.
- ✔ Python 2.0, introduced on October 16, 2000, brought forth numerous new features.
- ✔ Python 3.0, released on December 3, 2008,

❖ Python Features

These generate **.exe** files.
All the programming languages are not scripting languages.
There is the high maintenance cost.

Python is a versatile programming language known for its simplicity and readability. It offers a wide range of features that make it popular among developers. Here are some key features of the Python programming language:

1. Easy to Learn and Read: Python's syntax is straightforward and resembles the English language, making it easy for beginners to learn and write code.

2. Interpreted Language: Python is an interpreted language, which means you don't need to

SLTIET

compile your code before running it. This results in faster development cycles.

3. High-Level Language: Python is a high-level language that abstracts many low-level details, making it easier to focus on solving problems without getting bogged down in system-specific or memory management issues.

4. Cross-Platform: Python is available on various platforms, including Windows, macOS, and Linux. Code written in Python is usually platform-independent.

5. Rich Standard Library: Python comes with a comprehensive standard library that includes modules and packages for a wide range of tasks, such as file handling, regular expressions, networking, and more.

6. Dynamic Typing: Python uses dynamic typing, which means you don't need to declare variable types explicitly. The type of a variable is determined at runtime.

7. Object-Oriented: Python is an object-oriented programming (OOP) language, and it supports the creation and manipulation of objects, classes, and inheritance.

8. Support for Modules and Packages: Python allows you to organize your code into reusable modules and packages, promoting code reusability and maintainability.

9. Extensive Third-Party Libraries: Python has a vast ecosystem of third-party libraries and frameworks, such as NumPy, Pandas, Django, Flask, TensorFlow, and more, that extend its functionality for various domains like data science, web development, and machine learning.

10. Exception Handling: Python provides a robust mechanism for handling exceptions and errors, helping developers write more robust and reliable code.

11. Support for Multi-Paradigm Programming: Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming, giving developers flexibility in their coding approaches.

12. Garbage Collection: Python includes automatic memory management through garbage collection, which helps in efficient memory usage.

13. Community and Documentation: Python has a large and active community of developers, resulting in extensive documentation, tutorials, and a wealth of online resources for learning and problem-solving.

SLTIET

14. Open Source and Free: Python is open source and freely available, allowing developers to use and distribute it without licensing fees.

15. Integration Capabilities: Python can easily integrate with other programming languages like C, C++, and Java, enabling developers to leverage existing codebases.

16. Concurrency and Parallelism Support: Python offers various libraries and modules for concurrent and parallel programming, allowing developers to take advantage of multi-core processors and distributed computing environments.

These features, along with Python's versatility and readability, have contributed to its popularity in various fields, including web development, data analysis, scientific computing, machine learning, and automation.

❖ Python Applications

1) Development of Web Applications: Python is a capable language for crafting web applications. It offers libraries for managing internet protocols like HTML, XML, JSON, and handling tasks such as email processing and requests. For instance, Instagram relies on the Python web framework called Django. Python offers several valuable frameworks for web development, including:

- Django and Pyramid (Suitable for robust applications)
- Flask and Bottle (Micro-frameworks)

2) Creation of Desktop GUI Applications: Python supports Graphical User Interfaces (GUIs) for creating applications with user-friendly interfaces. Python includes the Tk GUI library, which facilitates the development of interactive user interfaces.

3) Console-Based Applications: These applications operate from the command-line or shell, executing commands to perform various tasks. Python is highly effective in developing console-based applications that rely on text-based interactions and command execution.

4) Software Development: Python serves as a valuable asset in the software development realm. It functions as a supportive language, aiding in the creation of control systems, management tools, and testing frameworks, among other uses.

5) Scientific and Numeric Computing: Python is exceptionally well-suited for fields like artificial intelligence and machine learning due to its rich array of scientific and mathematical libraries. These libraries simplify the handling of intricate calculations and data analysis.

6) Business Applications: Business-oriented applications, such as e-commerce and ERP systems, demand extensive functionality, scalability, and readability. Python delivers on all these fronts, making it an ideal choice for such applications.

SLTIET

7) Multimedia Applications: Python's versatility allows for the development of multimedia applications capable of handling audio and video tasks. Notable examples of such applications created with Python include TimPlayer and cplay.

8) 3D CAD Applications: Python possesses the capability to design 3D CAD (Computer-Aided Design) applications, which are instrumental in engineering and architectural endeavors. These applications enable the creation of detailed 3D representations.

9) Enterprise Solutions: Python can be harnessed to develop applications tailored for use within enterprises or organizations. Real-time examples of such applications include OpenERP, Tryton, and Picalo, offering a wide range of business functionalities.

10) Image Processing: Python boasts an array of libraries designed for image processing. These libraries empower developers to manipulate images to suit specific requirements. Notable examples of image processing libraries include OpenCV and Pillow.

Installing Python

There are many interpreters available freely to run Python scripts like IDLE (Integrated Development Environment) which is installed when you install the python software from http://python.org/downloads/

Steps to be followed and remembered:

Step 1: Select Version of Python to Install.
Step 2: Download Python Executable
Installer. Step 3: Run Executable Installer.
Step 4: Verify Python Was Installed On Windows.
Step 5: Verify Pip Was Installed.
Step 6: Add Python Path to Environment Variables (Optional)

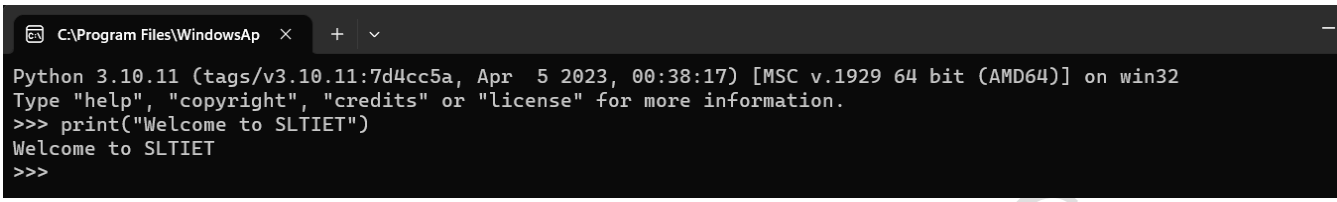
There are two modes for using the Python interpreter:

- ✔ Interactive Mode
- ✔ Script Mode

Running Python in interactive mode:

- ✔ Without passing python script file to the interpreter, directly execute code to Python prompt.
- ✔ Once you're inside the python interpreter, then you can start.
- ✔ >>> print("Welcome to SLTIET") Output:
  Welcome to SLTIET

SLTIET



Running Python in script mode:

- ✔ Alternatively, programmers can store Python script source code 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.

❖ Elements of Python

Python consists of four key elements:

1. IDLE (Python GUI): IDLE serves as a cross-platform Graphical User Interface (GUI) Integrated Development Environment (IDE) that comes bundled with Python. It offers a comprehensive suite of tools for Python development, including a Python Shell for script execution, a text editor for creating and editing Python source code, an integrated interactive debugger for error identification, and a help system, among others.

2. Module Documentation: This component enables access to Python's extensive documentation, encompassing built-in modules, DLLs, libraries, packages, and more. It serves as a valuable resource for developers seeking information on Python's capabilities and functionality.

3. Python (Command Line): Python is also accessible via the command line interface. While the command line mode offers fewer features compared to IDLE, it excels in terms of speed and simplicity, making it an efficient option for quick Python interactions.

4. Python Manual: This element encompasses a diverse array of documents related to Python, including installation and setup guides, tutorials, the Python API reference, frequently asked questions (FAQs), and other instructional materials. It serves as a comprehensive reference for users looking to explore various aspects of Python.

1.3 Basic Structure of Python program

The typical structure of a python program includes 3 parts:

SLTIET

1.4 Keywords and Identifiers

❖ Identifiers

✔ Identifier is a name given to various programming elements such as a variable, function, class, module or any other object.

✔ Following are the rules to create an identifier.

1. The allowed characters are a-z, A-Z, 0-9 and underscore (_)
2. It should begin with an alphabet or underscore
3. It should not be a keyword
4. It is case sensitive
5. No blank spaces are allowed.
6. It can be of any size

✔ Valid identifiers examples : si, rate_of_interest, student1, ageStudent

✔ Invalid identifier examples : rate of interest, 1student, @age

❖ Keywords

✔ Keywords are the identifiers which have a specific meaning in python, there are 33 keywords in python. These may vary from version to version

False	None	True	and
as	assert	break	class
continue	def	del	elif
else	except	finally	for
from	global	if	import
in	is	lambda	nonlocal
not	or	pass	raise
return	try	while	with
yield			

1.5 Data types and Variables:

❖ Variables

When we create a program, we often need store values so that it can be used in a program. We use variables to store data which can be manipulated by the computer program.

- ✔ Every variable has a name and memory location where it is stored.
- ✔ It Can be of any size
- ✔ Variable name Has allowed characters, which are a-z, A-Z, 0-9 and underscore (_)
- ✔ Variable name should begin with an alphabet or underscore
- ✔ Variable name should not be a keyword
- ✔ Variable name should be meaningful and short
- ✔ Generally, they are written in lower case letters
- ✔ A type or datatype which specify the nature of variable (what type of values can be stored in
- ✔ We can check the type of the variable by using type command

>>> type(variable_name)

Assigning Values to Variables:

- ✔ The equal sign (=) is used to assign values to variables.
- ✔ The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable.

For example –

```
a= 100 # An integer
assignment b = 1000.0 # A
floating point
c = "SLTIET" # A
string print (a)
print
(b)
print
(c)
```

Output:

```
100
1000.0
SLTIET
```

SLTIET

Multiple Assignments:

Python allows you to assign a single value to several variables simultaneously.
For example:
a = b = c = 1
Here, an integer object is created with the value 1, and all three variables are assigned to the same memory location. You can also assign multiple objects to multiple variables.

For example –

a,b,c = 1,2,"SLTIET“
Here, two integer objects with values 1 and 2 are assigned to variables a and b respectively, and one string object with the value "SLTIET" is assigned to the variable c.

❖ Data types

1. Number: Number data type stores Numerical Values. These are of three different types:

a) Integer & Long
b) Float/floating point
c) Complex

a) Integers are the whole numbers consisting of + or – sign like 100000, -99, 0, 17.
e.g. age=19
salary=20000
b) Floating Point are Numbers with fractions or decimal point. A floating point number will consist of sign (+,-) and a decimal sign(.)
e.g. temperature= -21.9,growth_rate= 0.98333328

The floating point numbers can be represented in scientific notation such as
-2.0X 105 will be represented as
-2.0e5 2.0X10-5 will be 2.0E-5
c) Complex: Complex number is made up of two floating point values, one each for real and imaginary part. For accessing different parts of a variable x we will use x.real and x.imag. Imaginary part of the number is represented by j instead of i, so 1+0j denotes zero imaginary part.
Example

```
>>> x = 1+0j
>>> print
x.real,x.imag 1.0
      0.0
```

2. Boolean: Objects of Boolean type may have one of two values, True or False:

```
>>> type(True)
<class 'bool'>
>>> type(False)
<class 'bool'>
```

3. Sequence: A sequence is an ordered collection of items, indexed by positive integers. Three types of sequence data type available in Python are Strings, Lists & Tuples.

SLTIET

a) String: is an ordered sequence of letters/characters. They are enclosed in single quotes (‘’) or double (“”).
Example
>>> a = 'CE'
>>>a=”CE”

b) Lists: List is also a sequence of values of any type. Values in the list are called elements /items. The items of list are accessed with the help of index (index start from 0). List is enclosed in square brackets.
Example
Student = [“SLTIETStudent”, 123, “CS”]

c) Tuples: Tuples are a sequence of values of any type, and are indexed by integers. They are immutable i.e. we cannot change the value of items of tuples . Tuples are written as(). Student = (“SLTIETStudent”, 123, “CS”)

4. Sets
Set is an unordered collection of values, of any type, with no duplicate entry. Sets are immutable.
Example
s = set ([1,2,3,4])

5. Dictionaries: Dictionaries store a key – value pairs, which are accessed using key. Dictionary is enclosed in curly brackets.
Example
d = {1:'a', 2:'b', 3:'c'}

6. String: Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes.

- ✔ 'hello' is the same as "hello".
- ✔ Strings can be output to screen using the print function.

For example: print("hello").
>>> print("Good Morning")
>>> type("Good Morning")
<class 'str'>
Literals:A literal is a constant that appear directly in a program and this value does not change during the program execution i.e. remain fixed.
Example:
Num1=5
Principle_amount=
5000.00

SLTIET

Here 5 and 5000.00 are literals. Python support the following literals

- ✔ Numeric literals
- ✔ String Literals
- ✔ Special Literals
- ✔ Collection Literals

❖ Comments:

Single-line comments begins with a hash(#) symbol and is useful in mentioning that the whole line should be considered as a comment until the end of line.
A Multi line comment is useful when we need to comment on many lines. In python, triple double quote(“ “ “) and single quote(‘ ‘ ‘)are used for multi-line commenting.

1.6 Type Casting

Type Casting is the method to convert the variable data type into a certain data type in order to the operation required to be performed by users.

There can be two types of Type Casting in Python –

- ✔ Implicit Type Casting
- ✔ Explicit Type Casting

Implicit Type Casting: In this, methods, Python converts data type into another data type automatically. In this process, users don’t have to involve in this process.

Example:

```
a = 7        #Python automatically converts a to int
print(type(
a))

b = 3.0      #Python automatically converts b to float
```

SLTIET

```
print(type(b))

c = a + b        #Python automatically converts c to float as it is a float
addition print(c)
print(type(c))
```

Output:

```
<class 'int'>
<class 'float'>
10.0
<class 'float'>
```

Explicit Type Casting: In this method, Python need users involvement to convert the variable data type into certain data type in order to the operation required.

Mainly in type casting can be done with these data type function:

- **Int() :** Int() function take **float or string** as an argument and return **int** type object.
- **float() :** float() function take **int or string** as an argument and return **float** type object.
- **str() :** str() function take **float or int** as an argument and return **string** type object.

Example:

```
a = 5
n = float(a)          # typecast to float
print(n)
print(type(n))
Output:
5.0
<class 'float'>
```

1.7 Input-Output functions: input, print

How to print/display a statement in Python

A statement is print with the help of print() method.

The syntax to use print() is as follow :

```
>>>print("message to be
printed") Or
>>>print('message to be
printed') Or
>>>print(variable_name)
```

Example:

```
x=10
print(
```

SLTIET

x) 10

Or

>>>print(“message to be printed”, variable_name)

Example:

```
>>x=10
>>print("value of x",x)
value of x 10
```

Or

>>>print(‘message to be printed’, variable_name)

Example:

```
>>x=10
>>print(‘value of x’,x)
value of x 10
```

How to input a value for the user in Python

A value can be input from the user with the help of input() method the syntax is as follow :

Syntax: variable_name = input(“Enter any number”)
input method return a string. It can be changed to other datatypes by using type.

Example:

```
age = int(input(“Enter your age”))
percentage= float(input(“Enter you percentage”))
```

1.8 Operators:

Operators are special symbols which represents specific operations. They are applied on operand(s),which can be values or variables.

✔ **Arithmetic Operators:** Arithmetic operators are used to apply arithmetic operation.

SLTIET

Arithmetic Operators in Python

There are 7 arithmetic operators in Python. The lists are given below:

Operator	Description	Syntax
+	Addition: adds two operands	x + y
–	Subtraction: subtracts two operands	x – y
*	Multiplication: multiplies two operands	x * y
/	Division (float): divides the first operand by the second	x / y
//	Division (floor): divides the first operand by the second	x // y
%	Modulus: returns the remainder when the first operand is divided by the second	x % y
**	Power (Exponent): Returns first raised to power second	x ** y

✔ **Relational Operators:** Relation operators are used to compare two items. The result of relational operator is true or false.

Relational Operators

Python	Mathematics	Meaning
<	<	Less than
<=	≤	Less than or equal to
==	=	Equal to
>=	≥	Greater than or equal to
>	>	Greater than
!=	≠	Not equal to

✔ **Logical Operators:** Logical Operators give the logical relationship based upon the truth table.

Logical Operators

Symbol	Description
or	If any one of the operand is true, then the condition becomes true.
and	If both the operands are true, then the condition becomes true.

✔ **Bitwise Operators :** Bitwise operators are applied upon the bits.

Bitwise AND operator: Returns 1 if both the bits are 1 else 0.
Example:

a = 10 = 1010 (Binary)

b = 4 = 0100 (Binary)

a & b =

1010

&

0100

= 0000

= 0 (Decimal)

Bitwise or operator: Returns 1 if either of the bit is 1 else 0.
Example:

a = 10 = 1010 (Binary)

b = 4 = 0100 (Binary)

a | b = 1010

|

01

00

= 1110

= 14 (Decimal)

Bitwise not operator: Returns one's complement of the number.
Example:

a = 10 = 1010 (Binary)

SLTIET

~a = ~1010

= -(1010 + 1)

= -(1011)

= -11 (Decimal)

Bitwise xor operator: Returns 1 if one of the bits is 1 and the other is 0 else returns false.
Example:
a = 10 = 1010 (Binary)

b = 4 = 0100 (Binary)

a ^ b = 1010

^

01

00

= 1110

= 14 (Decimal)

Bitwise left shift: Shifts the bits of the number to the left and fills 0 on right as a result.
Example

a = 5 = 0000 0101 (Binary)

a << 1 = 0000 1010 = 10

a << 2 = 0001 0100 = 20

Bitwise right shift: Shifts the bits of the number to the right and fills 0 on left(fills 1 in the case of a negative number) as a result.
Example:
a = 10 = 0000 1010 (Binary)

a >> 1 = 0000 0101 = 5

✔ **Membership Operator:** Membership operators are used to test if a sequence is presented in an object:

Example:1

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.

SLTIET

```
x = ["apple", "orange"]
print("apple" in x)
```

output:

true

Example:2

```
x = ["apple", "orange"]
print("pineapple" not in x)
```

output:

true

✔ Assignment Operators

Assignment operators are used to assign a value to the variable.

=	Assigned values from right side operands to left variable	>>>x=12* >>>y='greetings'	
*(*we will use it as initial value of x for following examples)*			
+=	added and assign back the result to left operand	>>>x+=2	The operand/ expression/ constant written on RHS of operator is will change the value of x to 14
-=	subtracted and assign back the result to left operand	x-=2	x will become 10
*=	multiplied and assign back the result to left operand	x*=2	x will become 24
/=	divided and assign back the result to left operand	x/=2	x will become 6
%=	taken modulus using two operands and assign the result to left operand	x%=2	x will become 0
**=	performed exponential (power) calculation on operators and assign value to the left operand	x**=2	x will become 144
//=	performed floor division on operators and assign value to the left operand	x / /= 2	x will become 6

SLIET