Unit 1 Fundamentals of Python

❖ Introduction to Python

- ✓ Python is a widely used general-purpose, high level programming language.
- ✓ Python is interpreted object-oriented language.
- ✓ It is open source software and easily available to use.
- ✓ It enables the straight and easy programming for both small and large applications. It mainly emphasizes on code reusability, readability and using white space.
- ✓ It has features like high-level built-in data structures, dynamic typing, and binding, which makes it attractive for rapid application development.
- ✓ Python is a programming language that lets you work quickly and integrate systems more efficiently.

❖ History of Python

- ✓ It was initially designed by **Guido van Rossum in 1991** and developed by Python Software Foundation.
- ✓ There are two major Python versions- **Python 2 and Python 3**.
- ✓ On 16 October 2000, Python 2.0 was released with many new features.
- ✓ On 3rd December 2008, Python 3.0 was released with more testing and includes new features.

❖ Differences between Scripting Language and Programming Language:

Scripting Language	Programming Language
A scripting language is a type of programming language designed for a runtime system to automate the execution of tasks.	A programming language is a computer language that is used to communicate with computers using a set of instructions.
It uses an interpreter to convert source code into machine code.	It uses a compiler to convert source code into machine code.
It is interpreted language or interpreter-based language	It is compiled language or compiler-based language
Examples include Perl, PHP,Python, JavaScript, etc.	Examples include C, C++, Java, Python, etc.
Execution of a script takes less time as scripts are generally short.	Execution of a program takes more time since they are compiled.
Do not create a **.exe** file.	These generate **.exe** files.
All the scripting languages are programming languages.	All the programming languages are not scripting languages.
There is less maintenance cost.	There is the high maintenance cost.

❖ Python Features

- ✓ **Python is object-oriented:** Structure supports such concepts as polymorphism, operation overloading and multiple inheritance.
- ✓ **Indentation:** Indentation is one of the greatest feature in python
- ✓ **It is free (open source):** Downloading python and installing python is free and easy.
- ✓ **It is Powerful:** It provides Dynamic typing, Built-in types and tools,Library utilities, and Automatic memory management

- ✓ **It is Portable:** Python runs virtually every major platform used today
- ✓ **It is easy to use and learn:** It has no intermediate compile. Python Programs are compiled automatically to an intermediate form called byte code, which the interpreter then reads.
- ✓ **Interpreted Language:** Python is processed at runtime by python Interpreter
- ✓ **Interactive Programming Language:**Users can interact with the python interpreter directly for writing the programs
- ✓ **Straight forward syntax:** The formation of python syntax is simple and straight forward which also makes it popular.

❖ Python Applications

1) Web Applications: We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, etc. One of Python web-framework named Django is used on Instagram. Python provides many useful frameworks, and these are given below:

- Django and Pyramid framework(Use for heavy applications)
- Flask and Bottle (Micro-framework)

2) Desktop GUI Applications: The GUI stands for the Graphical User Interface, which provides a smooth interaction to any application. Python provides a Tk GUI library to develop a user interface.

3) Console-based Application: Console-based applications run from the command-line or shell. These applications are computer program which are used commands to execute.Python can develop this kind of application very effectively.

4) Software Development: Python is useful for the software development process. It works as a support language and can be used to build control and management, testing, etc.

5) Scientific and Numeric: Python language is the most suitable language for Artificial intelligence or machine learning. It consists of many scientific and mathematical libraries, which makes easy to solve complex calculations.

6) Business Applications: E-commerce and ERP are an example of a business application. This kind of application requires extensively, scalability and readability, and Python provides all these features.

7) Audio or Video-based Applications : Python is flexible to perform multiple tasks and can be used to create multimedia applications. Some multimedia applications which are made by using Python are TimPlayer, cplay, etc.

8) 3D CAD Applications: The CAD (Computer-aided design) is used to design engineering related architecture. It is used to develop the 3D representation of a part of a system. Python can create a 3D CAD application

❖ 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)

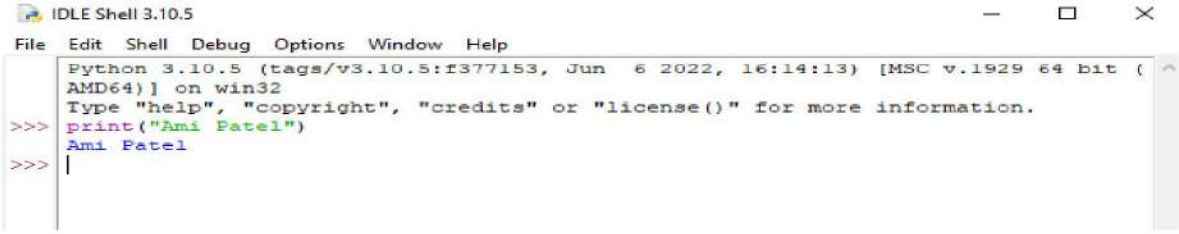
There are two modes for using the Python interpreter:

- ✓ 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("Ami Patel")

  Output: Ami Patel



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 has 4 components

1. **IDLE (Python GUI):** It is a cross platform GUI Integrated Development Environmental that is provided with Python for editing and running a Python programs. It is a bundled set of software's tools such as Python Shell for Scripting programs and text editor for creating and modifying Python's source code, an integrated interactive debugger for finding error, help system etc.
2. Module Docs: This component allows us to access the python documents such as build in modules, DLLs, libraries, packages etc.
3. **Python (Command line):** Python can also be access from the command line. Command line mode provide very less features in comparison to IDLE but it is fast.
4. **Python Manual :** This component include various documents related to Python such as : installation cum setup guide, tutorials, Python API , FAQs etc.

❖ Basic Structure of Python program

The typical structure of a python program include 3 parts:

Import statements // import statements are used to include library files to the python program
Function definitions //This section include the definitions of various functions written in a Python Program
Program statements // This section include the set of statements for solving the given problem.

❖ 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			

❖ **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 = "Neha" # A string
print (a)
print (b)
print (c)
```

Output:

```
100
1000.0
Neha
```

Multiple Assignment:
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,"Neha"
Here, two integer objects with values 1 and 2 are assigned to variables a and b respectively, and one string object with the value "Neha" 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: Numbers with fractions or decimal point are called floating point numbers.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.

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 = [“Dharmesh”, 567, “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 enclosed in ().
Student = (“Dharmesh”, 567, “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")
mrcet college
>>> 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
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.

❖ 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
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 user 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'>
```

❖ 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(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"))
```

❖ **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.

Symbol	Description	Example 1	Example 2
+	Addition	>>>55+45 100	>>> 'Good' + 'Morning' GoodMorning
-	Subtraction	>>>55-45 10	>>>30-80 -50
*	Multiplication	>>>55*45 2475	>>> 'Good'* 3 GoodGoodGood
/	Division	>>>17/5 3 >>>17/5.0 3.4 >>> 17.0/5 3.4	>>>28/3 9
%	Remainder/ Modulo	>>>17%5 2	>>> 23%2 1
**	Exponentiation	>>>2**3 8 >>>16**.5 4.0	>>>2**8 256
//	Integer Division	>>>7.0//2 3.0	>>>3/ / 2 1

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

Symbol	Description	Example 1	Example 2

<	Less than	>>>7<10
		True
		>>> 7<5
		False
		>>> 7<10<15
		True
		>>>7<10 and 10<15
		True
			>>>'Hello'< 'Goodbye'
			False
			>>>'Goodbye'< 'Hello'
			True

>	Greater than	>>>7>5 True >>>10<10 False	>>>'Hello'> 'Goodbye' True >>>'Goodbye'> 'Hello' False
<=	less than equal to	>>> 2<=5	>>>'Hello'<= 'Goodbye'
		True >>> 7<=4 False	False >>>'Goodbye' <= 'Hello' True
>=	greater than equal to	>>>10>=10 True >>>10>=12 False	>>>'Hello'>= 'Goodbye' True >>>'Goodbye' >= 'Hello' False
! =, <>	not equal to	>>>10!=11 True >>>10!=10 False	>>>'Hello'!= 'HELLO' True >>> 'Hello' != 'Hello' False
==	equal to	>>>10==10 True >>>10==11 False	>>>'Hello' == 'Hello' True >>>'Hello' == 'Good Bye' False

Note: Two values that are of different data type will never be equal to each other.

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

AND			OR			NOT	
Value1	Value2	Result	Value1	Value2	Result	Value1	Result
False	False	False	False	False	False	True	False
False	True	False	False	True	True	False	True
True	False	False	True	False	True		
True	True	True	True	True	True		

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.
not	Reverses the state of operand/condition.

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

OPERATOR	DESCRIPTION	SYNTAX
&	Bitwise AND	x & y
|	Bitwise OR	x | y
~	Bitwise NOT	~x
^	Bitwise XOR	x ^ y
>>	Bitwise right shift	x>>
<<	Bitwise left shift	x<<

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

|

0100

= 1110

= 14 (Decimal)

Bitwise not operator: Returns one's complement of the number.
Example:
a = 10 = 1010 (Binary)

~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

^

0100

= 1110

= 14 (Decimal)

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

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.

Example:1
```
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