UNIT-1

Python is also a **high-level language**; other high-level languages you might have heard of are C, C++, FORTRAN, PHP, and Java.

Also, we do have **low-level languages**, sometimes referred to as **machine languages** or **assembly languages**.

Machine language is the encoding of instructions in binary so that they can be directly executed by the computer.

Assembly language uses a slightly easier format to refer to the low level instructions.

Assembly language is also called as **mneumonic language**.

For every assembly language statement or instruction there will be one corresponding machine language statement.

First, it is much easier to program in a high-level language.

Programs written in a high-level language take less time to write, they are shorter and easier to read, understand, repair, or upgrade.

Second, high-level languages are **portable**, meaning that they can run on different kinds (both various hardware platforms or operating systems) of computers with few or no modifications. Low-level programs can run on only one kind of computer and have to be rewritten to run on another.

Two kinds of programs process high-level languages into low-level languages: **interpreters** and **compilers**.

An interpreter reads a high-level program and executes it, meaning that it does what the program says. It processes the program a little at a time, alternately reading lines and performing computations.

A compiler reads the program and translates it completely before the program starts running. In this case, the high-level program is called the **source code**, and the translated program is called the **object code** or the **executable**. Once a program is compiled, you can execute it repeatedly without further translation.

In the case of Python language, there are two ways to use the Python interpreter: *shell mode* and *program mode*. In shell mode, you type Python expressions into the **Python shell**, and the interpreter immediately shows the result.

Applications for Python

Python is used in many application domains. Here's a sampling.

- The Python Package Index lists thousands of third party modules for Python.

Web and Internet Development

Python offers many choices for web development:

- Frameworks such as Django and Pyramid.
- Micro-frameworks such as Flask and Bottle.
- Advanced content management systems such as Plone and django CMS.

Python's standard library supports many Internet protocols:

- HTML and XML
- JSON
- E-mail processing.
- Support for FTP, IMAP, and other Internet protocols.
- Easy-to-use socket interface.

And the Package Index has yet more libraries:

- Requests, a powerful HTTP client library.
- BeautifulSoup, an HTML parser that can handle all sorts of oddball HTML.
- Feedparser for parsing RSS/Atom feeds.
- Paramiko, implementing the SSH2 protocol.
- Twisted Python, a framework for asynchronous network programming.

Scientific and Numeric

Python is widely used in scientific and numeric computing:

- SciPy is a collection of packages for mathematics, science, and engineering.
- Pandas is a data analysis and modeling library.
- IPython is a powerful interactive shell that features easy editing and recording of a work session, and supports visualizations and parallel computing.
- The Software Carpentry Course teaches basic skills for scientific computing, running bootcamps and providing open-access teaching materials.

Installing Python on Windows, Mac, Linux

Go to the python website www.python.org and click on the 'Download' menu choice.

Next click on the Python 2.7 (note the version number may change) Windows Installer to download the installer. If you know you're running a 64-bit os, you can choose the x86-64 installer.

Be sure to save the file that you're downloading.

Once you've downloaded the file, open it. (You can also double-click on it to open it.)

You may get a warning that the file is executable. Just click ok at this prompt.

Once the installer starts, it will ask who to install the program for. Usually installing for all users is the best choice.

Next, it needs to know where to install the file. The default choice is fine.

You need to install the entire package

It will take a while to install.

Click 'Finish' to exit the installer.

After installed, you should now have a Python menu choice. Start the program by choosing IDLE (Python GUI)

This starts the python shell. You can type in simple commands to see how they work. Try typing the following:

4 + 4

print 'Hello world!'

In order to do more elaborate programs, normally people store all the commands in a file. To open a file to use in this way, go to File -> New Window.

Run your program, by going to Run -> Run Module.

The Python Shell

To start the shell under Windows simply go to the start menu and choose *Python (command line)* from the menu.

Any Python expression can be entered into the shell and you will see the result printed out for you right underneath

The >>> is called the **Python prompt**. The interpreter uses the prompt to indicate that it is ready for instructions. We typed 2 + 3. The interpreter evaluated our

expression and replied 5. On the next line it gave a new prompt indicating that it is ready for more input.

Working directly in the interpreter is convenient for testing short bits of code because you get immediate feedback.

Elementary Input/Output in Python

Let us first try to write a simple Python program which greets to the user with the following messages when executed.

Hello Welcome to Interactive Python Learning

Hello Welcome to Interactive Python Learning

For this purpose, we need to use print function. Like majority of programming languages, in Python also any text enclosed between double quotes is called as string literal or constant. Whatever message we want Python to display on the console should be enclosed between double quotes and used with print function in either of the following ways.

print "Hello Welcome to Interactive Python Learning"

print("Hello Welcome to Interactive Python Learning")

The most prominent difference between these two Python versions is that in Python 2.x, the print command is special where as in Python 3 it is an ordinary function. That is, in Python 2, both the two versions illustrated above can be used with print function while in Python 3, only the one with paranthesis is acceptable.

print 'Hello Welcome to Interactive Python Learning'

print "'Hello Welcome to Interactive Python Learning'"

print """Hello Welcome to Interactive Python Learning"""

Variables and Types

Python is completely object oriented, and not "statically typed". You do not need to declare variables before using them, or declare their type. Every variable in Python is an object.

This tutorial will go over a few basic types of variables.

Numbers

Python supports two types of numbers - integers and floating point numbers

```
myint = 7

print(myint)
```

Strings

Strings are defined either with a single quote or a double quotes.

```
mystring = 'hello'

print(mystring)

mystring = "hello"

print(mystring)
```

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

Indenting Code

Python functions have no explicit begin or end, and no curly braces to mark where the function code starts and stops. The only delimiter is a colon (:) and the indentation of the code itself.

Code blocks are defined by their indentation. By "code block", I mean functions, if statements, for loops, while loops, and so forth. Indenting starts a block and unindenting ends it. There are no explicit braces, brackets, or keywords. This means that whitespace is significant, and must be consistent

How To Give Interactive Input in Python?

Now, let us try to write a simple Python program which reads a persons name and greats him. In order to do this, we need a way to get **input** from the user. In Python there is a built-in function to accomplish this task, which is called input.

n = input("Please enter your name: ")

print("Hello ", n, ", Welcome to learn python interactively")

The input function allows the user to provide a **prompt string**. When the function is evaluated, the prompt is shown. The user of the program can enter the name and press *return*. When this happens the text that has been entered is returned from the *input* function, and in this case assigned to the **variable** *n*

In Python, we can use plus(+) operator (also known as string concatenation operator) between two strings.

```
x="Abdul"

y="Kalam"

z="Dr. APJ"

print(z," ", x, " ", y)

print(z+x)

print(z+x+y)


x,y,z=input("Enter Surname Middle Last names")

print z,x,y

print(z+x+y)

x="Abdul"

y="Kalam"

z="Dr. APJ"

z

x

y

z,x,y
```

```
z+x+y

p=z,x,y

print p
```

We can also use another input function known as raw_input(). This function takes input from the user and return the same as a string. If we want, we can split this string into values using split() method. For Example, the following program demonstrates the same.

```
x,y,z=raw_input("Enter Surname Middle Last name").split()

print z,x,y

print(z+x+y)


x,y,z=raw_input("Enter Surname Middle Last name").split(',')

print(type(x))

print z,x,y

print(z+x+y)


x,y,z=input("Enter three integers separated by spaces").split()

#x,y,z=int(x),int(y),int(z)

print z,x,y

print(z+x+y)


day,mon,year=input("Enter date like 17-12-2018").split("-")

day,mon,year=int(day),int(mon),int(year)

print(day, mon, year)
```

Values and Data Types

Python has a function called **type** which can tell inform what class an object or value is.

print(type("Hello, World!"))

print(type(17))

When we run the above program, we get output as <type ‘str’> and <type ‘int’> indicating the first value “Hello, World!” is **str** (string) type while 17 is **int** (integer) type.

Similarly, numbers with a decimal point belong to a class called **float**, because these numbers are represented in a format called *floating-point*.

print(type("17"))

print(type("3.2"))

print(42000)

print(42,000)

Because of the comma, Python chose to treat this as a *pair* of values. In fact, the print function can print any number of values as long as you separate them by commas. Notice that the values are separated by spaces when pythonthey are displayed.

print 42,37,56,34

print 3.4, "Hello", 45

print(42, 17, 56, 34, 11, 4.35, 32)

print(3.4, "Hello", 45)

Type Conversion

Sometimes it is necessary to convert values from one type to another. Python provides a few simple functions that will allow us to do that. The functions int,

float and str will (attempt to) convert their arguments into types *int*, *float* and *str* respectively. We call these **type conversion** functions.

The int function can take a floating point number or a string, and turn it into an int. For floating point numbers, it *discards* the decimal portion of the number

```
print(3.14, int(3.14))
print(3.9999, int(3.9999))       # This doesn't round to the closest int!
print(3.0, int(3.0))
print(-3.999, int(-3.999))       # Note that the result is closer to zero

print("2345", int("2345"))       # parse a string to produce an int
print(17, int(17))               # int even works on integers
print(int("23bottles"))
```

The type converter float can turn an integer, a float, or a syntactically legal string into a float.

```
print(float("123.45"))
print(type(float("123.45")))
```

The type converter str turns its argument into a string.

```
print(str(17))
x=str(17)
print(x,type(x),type(17))
```

```
print(str(123.45))

y=str(123.45)

print(y,type(y),type(123.45))

print(type(str(123.45)))
```

The following program reads initial velocity,time duration then displays distance travelled.

```
u = int(raw_input("Enter initial velocity in m/sec"))

print(type(u));

vel = int(u)

print(type(vel))

t = int(raw_input("Enter time travelled(sec)"))

a = int(raw_input("Enter accelaration(m/sec^2)"))

print(type(t))

distance=vel*t + 0.5*a*t*t;

print("Distance Travelled=", distance)
```

```
import math


num=float(input("Enter a real number"))
print(num, math.ceil(num),math.floor(num))
```

Write a program to compute hypotnus of a right angled triangle given base and height of the triangle.

```
import math


a=float(input("Enter base of a triangle"))
b=float(input("Enter height of a triangle"))
h=math.sqrt(a**2+b**2)
print "Hypoteneus=", h
```