Introduction to Python

Chapter 1: Overview of Python

Part 1: Introduction to Python

Slide 1

Welcome to our course on Python. In this module, we will introduce Python and tell what it can do for you. Python, as you know, is a high-level programming language. It makes you work faster. You can deliver code faster and integrate systems more effectively than most other languages. We will also talk about how you can make the best use of Python to meet your programming needs. We will talk about the data types, data structures, and language constructs, such as decision making statements and repeat constructs. We will also give an overview of concepts of using strings, tuples, sets, iterations and loopings, objects, generators, and dictionaries. Python has so much to offer you. We could just go on and on explaining what Python can do for you.

It's our goal to make you conversant with the features of Python. Once you go through the series of modules, you can comfortably write programs in Python. You should be able to develop and deliver complex applications with ease. We will be showing a few demos, as we go through to show you how the features really work. It is just not that. We strongly recommend that you practice those on your own. We are sure you will be excited to know more about Python.

Welcome to the world of Python.

Slide 2

Let’s start with features offered by Python. In subsequent modules, we will dwell into each of the major features of Python. But first, we’ll have a bird’s eye view of Python. Python is a high-level language, meaning that it is not like an assembly level language. This means that you don’t need to compile the code to get an executable and then use it. Python is an interactive language. And it supports object-oriented features. It is as simple as Perl and PHP, but has its own unique features.

I want to reiterate the key features of Python. Python was designed to be highly readable. It often uses English keywords. Python is interpreted which is actually an advantage that it can be processed at runtime by the interpreter. You don’t need to compile your program before executing it. You can actually sit at a Python prompt and interact with the interpreter directly to write your scripts. That’s why I say it is highly interactive.

You will soon realize that Python has efficient high-level data structures. Yes, they are simple but isn’t it interesting to dig deeper into data structures? The approach to object-oriented programming is very effective in Python. Python is a beginner's language for programmers because of its simplicity, which makes it similar to Perl and PHP. We hope that helps you understand just how simple it is. I want to say here that Python has a wide range of applications from simple text processing to web browsers to games, and from simple to complex programs.

Slide 3

Before we go any further, let’s talk little bit about the history of Python. Knowing the history generally gives us the

right perspective. It came out to make life easy, but to still provide the best of all the then available programming languages.

Python was developed by Guido van Rossum. He was with the National Research Institute in Netherlands. Rossum holds the copyrights to Python and leads the developments to Python language. Even though it is copyrighted, it's free. Python source code is under the GNU General Public License, GPL. A core development team is formed at the institute to manage the development and progress on Python. As we said before, Python is derived from many other programming languages available at the time of its development, including C and C++.

Slide 4

We just mentioned that Python is free and is under GNU. But where do you get it from? The latest versions, source code, binaries, news, documentation, etc., are now available for download from its official website: http://www.python.org/. The environment could be Unix, a variant of Unix, Windows, Mac OS X, etc. You can also download the documentation. The documentation is available in HTML, PDF, and PostScript formats. It helps you enhance your skills and refer to when you need. The website for Python documentation is www.python.org/doc/.

As we said, Python distribution is available for a wide variety of platforms. We strongly recommend you download Python binary, not the source code for your platform, install and start using it. If the binary code for your platform is not available or you face glitches, then you should download the source code and compile it on your own. You may feel it's a little difficult but

compiling Python on your own offers you more flexibility in terms of choice of features that you require for your installation. Remember that my recommendation is to first download the binaries.

Part 2: Features of Python

Slide 5

We briefly highlighted the key features of Python, but we didn't really talk about the language constructs. So, now we are going to talk about some of the most important language features and constructs of Python. Since this module is introducing Python, we will be brief. The details of the language will be covered in subsequent modules.

As we've mentioned, Python is easy to learn. It is a beginners' language, having relatively few keywords and a simple structure. Moreover, it has a clearly defined syntax. It will not take long for programmers to learn and start writing programs using Python. One of the key success factors for Python is that its source code is quite easy to maintain. This means the program meets your current needs by updating it suitably, may be by someone else. Python code is extremely easy to understand and maintain.

Python is a minimalistic language. People call it a pseudocode natural language. This is yet another strength of Python. Rather than worrying about language, syntax, and the nitty-gritty of the language, it helps us concentrate on the solution to the problem. That makes it simple and user friendly.

Python is a classic example of a free/libre/open-source software (FLOSS). I'll explain: You can freely distribute copies of this software. You can obtain and read its source code. You can make changes to it. You can use it in any other programs. FLOSS is based on the concept of a community which shares knowledge. This is also a reason why Python is widely used. By the way, there are both 32 and 64-bit versions for most operating systems.

Python supports procedure-oriented programming as well as object-oriented programming. What does that mean? Most programmers in their early weeks or months of programming life try to figure out what these are. They find it a little difficult to switch between the thought processes behind these. In procedure-oriented programming, the programs are built around procedures or functions. These are nothing but reusable pieces of programs. This is how you modularize the code. You make it readable and easy to maintain. In an object-oriented approach, the program is built around objects which combine data and functionality. Python is very powerful in this approach. However, it has a simpler way to implement it, in comparison to other languages such as C, C++, and Java.

Python is dynamically typed. This is same as =Perl. This means variables do not need to be declared as real, integer, string, etc., at first. You can start using them straightaway unless you have a reason to define them early. The type of variable is automatically defined by the value assigned to it. So, the value decides the type of variable. The type decides the operations you can perform on a variable. These are primarily decided at

runtime as Python is an interpreted language and does not have an executable version.

If Python is a dynamic typed language, what are static typed languages? You have veteran languages such as Fortran and C. They have the data types defined upfront and have an executable. The type and the operations are decided at compile time.

Python has an extensible feature. You can import modules from other languages into Python. You don't need to redevelop the code in Python if there is a piece of code available in another language and if it can be imported.

It's not only extensible. In fact, you can embed Python code in C and C++ programs to provide better scripting capabilities there.

Python has a huge standard library. The library contains many packages that can help us to do various things involving regular expressions, databases, documentation generation, and other system-dependent stuff.

Python is also portable. Its open-source nature is its biggest strength. Generally, Python programs can work on any platform without requiring us to make any changes. You can port Python programs with Linux, Windows, Mac, OS, OS/2, Solaris, PlayStation, Windows CE, PocketPC, etc.

We keep saying that Python is an interpreted programming language. We also briefly mentioned some of the differences from other languages. You may ask, what is the difference between an interpreted language and a compiled language? Languages such as Fortran, C, and C++ are compiled languages. The entire source

code file is read and then converted into an executable code, or a machine level language. This executable will have many operating system or platform dependent instructions, which may not work on other operating systems or platforms. Data types and operations on them are decided upfront. It can be executed directly on that particular operating system.

In an interpreted language, the source code text file is read line-by-line and each statement is converted into machine language and then executed before the next line is read. There are no two steps of compilation and execution. In the case of compiled languages, it is compiled once and used multiple times. In interpreted languages the programs are converted each time you run it. Generally, compiled languages are more efficient and faster because you don't need to convert the code to machine language. It comes already in machine language, and in most cases, in optimized form. However, interpreted languages are more flexible and changes can be easily done to the program at runtime.

Having seen some of the key features of Python, let's see what's new in Python 3.4.

Slide 6

We'll start with the bootstrapping of pip in Python applications.

The new ensurepip module provides a regular cross-platform mechanism to bootstrap the pip installer into Python installations and virtual environments. Note that I said "virtual" environments. Python is not lagging behind the rest of

the IT world in terms of adapting the latest technological developments. Commands such as pipX and pipX.Y normally come along with the Python package. Here X.Y denotes the version of the Python installation.

In Python 3.4, the documentation has been reframed for installing and distributing Python modules. This work is still in progress and legacy versions are still available.

Let me talk about the next key change. Newly created file descriptors are now non-inheritable in Python 3.4. This is what an application would like to have when you launch a new process. Allowing currently opened files to continue to be open when a new process is launched can lead to bugs which programmers may find difficult to locate. However, this cannot be made mandatory. There are cases where you want the inheritance to work. Python 3.4 provides new functions to handle this by making use of the following functions and methods: os.get_inheritable, os.set_inheritable, os.get_handle_inheritable, os.set_handle_inheritable, socket.socket.get_inheritable, and socket.socket.set_inheritable

There are improvements to codec handling. Type-neutral dynamic encoding and decoding systems are intentionally operated by the codecs module. The codecs.encode and codecs.decode functions are listed out in documentations of Python's earlier versions. In the codecs module, these functions have been in use since Python 2.4. In earlier versions, they are discovered only through runtime introspection.

Now there is a ModuleSpec type for the Import System. Import implementation as well as import linked APIs are simplified

using ModuleSpec type. This transformation creates a way in for several import related improvements in the future. Backward compatibility is a predominant problem that developers face. Those types are transparent. This may not be true for importer authors. Some methods such as Key Finder and Loader are removed but still work. Importers who are new can use the new methods that are described in PEP while existing importers can upgrade to implement the latest methods.

Part 3: Python Development Environment

Slide 7

So far we have talked a lot about the features of Python. Now we're going to move on to the environment required for Python to be installed and used. Before we proceed further, let's look at versions of Python. The stable versions of Python are Python 2.7.8 and Python 3.4.1. Python 3.5 is under development at the time of recording this module.

Python 3.0 does not maintain backward compatibility with the older versions of Python. In other words, code developed for Python 2.x may not work with Python 3.x, and vice-versa. Because much code and so many libraries have been developed for Python 2.x, most of the programmers in the scientific community still use 2.x.

Python supports both 32 and 64-bit server installation. In Windows, it supports for Windows XP or later. In Unix/Linux, it supports Fedora, Debian, OpenSuse, Slackware, and Ubuntu. Memory requirements for Python are very small; we're talking about GBs

of memory. Python is content with 512MB RAM, but I personally recommend 2 GB for better performance.

Python is somewhat similar to Java in terms of extending to different platforms. It runs on many platforms like Windows, Linux, Mac OS X, OS/2, Amiga, Palm Handhelds, and even on Nokia mobile phones. It has an active open source development that innovates on a continuous basis.

Slide 8

As a developer you would want an integrated development environment (IDE). You would not want to program in native mode. Let's see some of the highly rated IDEs of Python. We will have a demo on how to install and use one of them.

Komodo IDE and Edit is a full featured developed environment GUI. Komodo has features such as syntax coloring, text editing, debugging, etc. Komodo offers advanced features such as project files, source-control integration, and regular-expression debugging. You can try this out during the 21-day free trial.

Eclipse PyDev is an advanced open source IDE GUI. Eclipse is a multi-level language supporting platform. It is very popular among developers and on campuses. It was developed as a Java IDE. Eclipse supports Python development only when you install the PyDev plugin. It offers some features such as code completion, syntax highlighting, syntax analysis, refactoring, debugging, etc.

NetBeans IDE is a community driven effort. NetBeans is a powerful open source development environment GUI. NetBeans advanced features for Python are code completion, automatic

indentation and code colorization, editor hints, code folding, refactoring, debugging, code coverage and testing, projects, and more. NetBeans develops both CPython and Jython code. NetBeans requires installation steps similar to Eclipse; however, Oracle does not actively play a part in it. In other words, Oracle won't directly support this effort.

Wingware's Python IDE, WingIDE, is a commercially created programming language for Python. It supports both versions of Python 2.x and 3.x. Also it can be used with Django, matplotlib, Zope, Plone, App Engine, PyQt, PySide, wxPython, PyGTK, TkInter, mod_wsgi, pygame, Maya, MotionBuilder, NUKE, Blender, and many other Python frameworks.

Boa Constructor is a cross platform Python IDE and wxPython GUI Builder. It offers you visual frame creation and manipulation, an advanced debugger, integrated help, etc.

Stani's Python Editor, SPE, is also a cross-platform IDE for Python developed and managed by Stani Michiels. SPE is a free software that runs on GNU/Linux, Mac OS X, and Microsoft Windows. It is based on wxPython. SPE is available under GNU General Public License. Some features of SPE are auto indentation and completion, call tips, syntax coloring and highlighting, UML viewer, class explorer, source index, to do list, PyCrust shell, file browsers, drag and drop, and Blender support. SPE ships with wxGlade, PyChecker, and Kiki.

Python comes with a nice interactive development environment. IDLE has been wrapped together with the default implementation of Python language. It is an optional part of Python and with many Linux distributions. It is written in Python and TkInter

GUI toolkit. IDLE is a beginners' IDE, especially for educational purposes.

In the demo we will be using IDLE and will also show how to install a Python plugin in NetBeans IDE.

Slide 15

There are three ways to run a Python program and execute it. Python programs are executed by an interpreter.

If Python is installed on your computer, you can enter the program on the command prompt. Then, you can start coding right away in the interactive interpreter by starting it from the command line. You can do this on any platform that provides command-line interpreter or shell window. For Unix/Linux, use $python or python%; in Windows/DOS, use C:>python. There are some built in command-line options to perform operations.

The second way is by using command line. A Python script can be executed at command line by invoking the interpreter on your application. For Unix/Linux, you should use $python script.py or python% script.py. For Windows/DOS, C:>python script.py should be used. Always be sure that the file permission mode allows you to execute.

The third way is by using an Integrated Development Environment. This is the most sought after means to develop and execute Python programs. You can run Python from a graphical user interface, GUI, environment. It's possible by just using a GUI application that supports your environment.

Slide 16

I would like to show you how to run your first Python program. I will use interactive mode to demo this. I have created a file with name helloworld.py. Note that Python programs end with the extension .py. I wrote a simple script to display "Hello World" in the prompt window. I just used print statement to print the output. To execute the helloworld.py file, I provided the filename to the interpreter as follows: %python helloworld.py. It automatically executes the file and displays the output as "Hello World". This is the simplest and easiest way to execute a Python program. Python programs can also be executed on Unix. You can use #! on the first line of the program. It displays the same output. The interpreter runs statements until it reaches the end of the input file. If it's running interactively, you can exit it. To exit the interpreter, type the EOF (end of file) character or select Exit from the pull-down menu of a Python IDE. In Unix, you can use Ctrl+D. For Windows, it's Ctrl+Z to exit.

Part 4: DEMO: Python

Hello, welcome to our demo for Python. Before we get into the demo, we're going to find out what version is available by typing python hyphen capital V. If it is a small v, it will show an error. It shows the version is Python 3.4.1.

You can get help by typing python hyphen h.

There are many other options, like hyphen V for print the version. You also have other environment variables, such as PYTHONSTARTUP, PYTHONPATH, PYTHONHOME, PYTHONCASEOK, PYTHONENCODING, PYTHONFAULTHANDLER, and PYTHONHASHSEED.

We will able to see how they have been set and how you can make necessary changes.

Now, let's run hello world python program and see the output, but before we do that, let me show you the source code for this program. The source code says helloworld.py. Now we are going to execute this program from the command prompt. Type python helloworld.py and the output is "Hello World."

Let's run another program, another version of Python. Here is the sys.version, let's see the output of this program. Go back to the command prompt and type python hello.py Here it shows, "hello from Python 3.4.1 with full version of Python details. This is how you can send data out and get it.

Now I can run this in IDLE itself. Go to Run menu and click Run Module. I can get the same output here, "Hello from Python", and it shows the version of python with all the details. This is the two ways you can do this, one from the command prompt and the other from IDLE.

I hope you have enjoyed seeing the running of your first Python program and its output.

[end demo]

Chapter 2: Functions of Python

Part 1: Data Types and Operators in Python

Slide 17

Identifiers and reserved words are basic building blocks of any programming language. An identifier is used to identify a

module, function, variable, class, or any other objects. You probably noticed that I used the terms function, class, module, and object. We will be seeing each of these in the subsequent modules.

Let's get started with the identifiers of variables. In its simplest usage, an identifier is a name given to a variable. We're going to work with that definition for now, though we can also use it for functions, classes, objects, etc.

An identifier name starts with an uppercase or lowercase letter, A to Z, or an underscore. They can be followed by numerals, 0 to 9, letters, or an underscore. We should not use any punctuation or symbol characters, such as at, dollar, and percent, within identifiers as it causes an error. Python is a case sensitive programming language.

Here's a list a few naming conventions for Python. Class names should start with an uppercase letter and all other identifiers with a lowercase letter. This is only a convention to help us understand and maintain code better. If an identifier starts with a single leading underscore then it indicates that the identifier is meant to be private. If it is started with two leading underscores, it indicates a strongly private identifier. Ending an identifier with two trailing underscores means that it is a language-defined special name.

Next comes the reserved words. Reserved words should not be used as constants, variables, or identifiers. They should be in lowercase letters only. Some of the reserved words are: and,

exec, not, assert, finally, or, break, for, pass, class, from, print, continue, global, raise, def, if, and return.

We will see how we use reserved words and identifiers later. We're now going to look at the syntax of a Python program. I will also explain the syntax for a few language constructs in a demo.

Slide 18

As we said, Python is a simple language. It has no brackets, braces, or parentheses to mark blocks of code, code for classes, code for functions, etc. This could make life difficult for a programmer. Python has a different approach to address this problem.

Quite simply, Python relies heavily on indentation. Blocks of code are denoted by line indentation and without braces. These indentations are rigidly enforced. The number of spaces in the indentation should be consistent in block of a code. Statements in Python generally end with a new line. Python, however, uses the line-continuation character backward slash to denote that the line is continuing. Statements within square brackets, curly braces, or parentheses need not to use line-continuation characters to start a new line. Python allows multiple statements to be on a single line separated by a semicolon. Python allows single, double, and triple quotes to denote string literals. Remember to use the same type of quote to start and end the string. Triple quotes are used to extend the string across multiple lines.

A hash sign that is not used inside a string literal is considered a comment by Python. The interpreter ignores and takes all the characters as a comment that is used after a hash to the physical line. Python allows command-line arguments. It means you can provide inputs via command line, making it interactive.

Slide 19

We're going to move on to data types in Python. We talked about identifiers and reserved words. So, we are back to identifiers, or names of variables in this case. In Python, variables are reserved memory locations that are used to store values. Whenever you create a variable, some spaces are reserved for that in the memory. Actually, the interpreter allocates memory based on the datatype of a variable specified. It also decides what can be stored in the reserved memory. By assigning different data types to variables, you can store integers, decimals, or characters. When you assign a value to a variable, the declaration is done automatically. You don't need to have an explicit declaration of a variable in Python.

How do you assign values to variables? You can do this using an equal sign. In an assignment statement, the operand to the left of the equals is the name of the variable and to the right is the value stored in that variable. Python also allows you to assign a single value to multiple variables simultaneously. We'll see more about standard data types in the next slide.

Slide 20

A classic example of standard data types is a person's age needs to be stored as a numeric value and his or her address needs to be stored as alphanumeric characters. Python has five standard data types. They are used to define the operations and the storage for each of them. These data types are: Numbers, String, List, Tuple, and Dictionary. Set is a variant of list. We will look at each of these types in subsequent slides.

Slide 21

We just saw that there are five basic data types supported in Python. Data types tell us about what kind of values can be stored in a variable and which operations can be performed on each of them.

The Python language has an extensive set of operators. They are classified into different categories. There is also an order of preference in which order the operators get used in the evaluation of expressions.

Python has arithmetic operators to carry out addition, subtraction, multiplication, division, exponentiation, floor division, modulus, etc.

It has comparison operators to compare the operands. This could be to check equality, inequality, greater than, greater than or equal to, less than, or less than or equal to.

It has assignment operators combined with standard arithmetic operators. For example, a+=b is same as a= a+b.

Python also has logical operators such as AND, OR, and NOT. In addition, Python has bitwise operators. It has membership

operators to check if something is a member of a sequence or not. There are two membership operators, in and not in. They are used to find out whether a variable is used in the specified sequence or not.

Python has identity operators to compare the memory locations of two objects. There are two identity operators, is and is not.

In Python, data can be converted from one type to another. This is a key point to remember.

Slide 22

Now we're going to get to know a little more about Number and String data types. Number data types store numeric values. They are immutable data types. Number objects are created as soon as you assign a value to them. You can use the del statement to delete the reference to a number. Python supports four different number types. Int for signed integers and long for larger integers. You also have octal and hexadecimal representation. Float is for floating point real values, and complex for storing complex numbers. You should use a lowercase "L" for declaring long integer variables because sometimes it leads to confusion with the numeral one. A complex number consists of an ordered pair of real floating-point numbers. It is denoted by a + bJ, where a is the real part and b is the imaginary part of the complex number. Python supports mathematical and trigonometric functions such as abs, ceil, cmp, exp, fabs, floor, log, max, min, modf, pow, round, sqrt, sin, asin, and hypot.

Strings are a set of characters in between quotation marks. Python does not directly support character types. A character is

treated as a string of length one, as I mentioned earlier. You can perform operations similar to other languages on strings. Python allows for either pairs of single or double quotes. Subsets of strings can be used by slice operators ([ ] and [:]) with indexes starting at 0 and -1 at the end. In Python, the plus sign is the string concatenation operator and the asterisk is the repetition operator.

What have we seen so far on data types? In Python, Number data types store numeric values. Strings are stored as a sequence of characters with a starting index of zero. Strings can be enclosed in single, double, and triple quotes. Triple quotes are used for multiple lines.

Part 2: DEMO: Assigning Values to Variables

Let's look at how Python assigns values to variables. We understood the fact that we can have multiple assignment in a single line.

Let’s look at how this is done. Also, we said that Python is contextual sensitive. We don't need to specify the data type upfront. Python will assign data type and storage appropriately. Python assigns 100 to counter. It is an integer. So, it will show integer later. Miles will be 1000.0, which means that it stores a float/real number and name is given a value of John. So, it takes it as a string variable.

Remember that in Python we don't have a data type. It is taken as a string of length 1. You can print the variable.

Also, we talked about, Assign values to multiple Variables in the same statement. Here, all the variables will have the same

value. Here, var1, var2, var3 are assigned the value 90. Let us print each variable and see each will be 90.

It's not necessary that, you have to have the same value in all the cases. We can assign different types of values and different types of different variables in the same statement.

When we do this, var1 has the value 80 and var2 has the value 92.75, and var3 has the value "john". Three different values with three different variables. Var3 is a string variable, var2 is a real/float number, and var1 is a number.

Let us run and let us see how the output is. To run the program from IDLE, select Run in the menu and then click on the Run Module.

You can see the output here. In the first case var1, var2, and var3 all have the value of 90. In the second case var1 is 80, var2 is 92.75, and var3 is john. This is how you can set different values to different variables of the same time.

We have quite a lot of functions available, let us import math module here so that we will be able to perform various math functions.

Let’s start with this, one function is here, abs(-45), must get the value 45. Now, to find the smallest integer not less than the value, ceil(-45.17). To get the exponential value of -45.17 to get absolute value -45.17.

You can also get largest integer floor of a particular real number then you also find out the largest of its arguments and smallest of its arguments. And also find out exponential of

number, for example, I can give 100 to the power of 2 should give me 10,000. Let's run this to find out what the output.

Let’s go to the run menu and click on the "Run Module".

Now, you get the absolute value of -45 is 45, Ceiling of -45.17 is -45 and exponential of -45.17, we can see how it shows 2.415006, and the absolute value of -45.17 is 45.17. Floor value of -45.17 is -46 is here. Note the floor for negative number. Then we have maximum of 80,100, 1000 is 1000 and minimum is 80. And then the power of 100 to the power of 2 is 10,000. And, the rounding option up to 3 decimals, then we have, square root of 100 is 10. This is how we do various mathematical functions.

[end demo]

Part 3: Lists, Tuples, and Dictionaries

Slide 23

We have seen the numbers and strings. Now we’ll talk about lists, tuples, and dictionaries. Lists are sequences of arbitrary objects. You can create a list by enclosing values in square brackets. Lists are indexed by integers, starting with zero as the first index. We can use the indexing operator to access and modify individual items off the list. We can use the append method to append new items to the end of a list. We can use the insert method to insert an item into the middle of a list. Python allows you to reassign a portion of a list by using the slicing operator. Use the plus operator to concatenate lists. Lists may contain any objects, including other lists.

We can use lists and tuples together to represent data. Tuples are the same as lists except that the data cannot be modified. Tuples can be created by just enclosing a group of values in parentheses. Often Python recognizes that a tuple is intended even if the parentheses are missing. Tuples and lists perform some operations almost the same, such as indexing, slicing, and concatenation. In fact, there is so much overlap in using tuples that some programmers find it more convenient to use lists other than tuples.

A dictionary in Python is an associative array or hash table that contains objects indexed by keys. A dictionary can be created by enclosing the values in curly braces. This is a useful way to define an object that consists of named fields. However, dictionaries are also used as a container for performing fast lookups on unordered data. Although strings are the most common type of key for a dictionary, but we can also use numbers and tuples. Some objects, including lists and dictionaries, cannot be used as keys because their contents can change.

I think we covered enough information on identifiers to get going with the next sections.

Part 4: DEMO: String Variables

In this demo we are going to look at string variables and how they are processed.

Let's assign "Hello World!" to string str. And see what happens when we print. Then to print the first character using an index, use index of zero. You can print the fifth character and sixth

character. of the string, it will print that way. Then we can also do the string slicing, what I mean to say here is. Suppose if we want to print the third element to the fifth character, I can specify, the index of the starting element and up to the fifth character. str[2:5], it will print “llo”.

Let us see in this example, we have print starting from the third character onwards. Here, it is str[2:],the starting index is given, there is no end. So, it prints till the end of the string. One more operator called star (*) operator, which is known as a repetition operator. So, I can print two times, three times or four times etc. Whatever is in string here is repeated two times. So “Hello World!” is repeated two times. Now let us see concatenate operator, plus (+) that operator concatenated with “Hello World!” gets appended with "TEST" in the end.

Let us run the program and see the output. Hello World, H is the first character, llo is the third, fourth, and fifth characters. Starting from the third element to the end you have “llo World”. Then you have “Hello World” repeating twice, and you have Hello world with the TEST concatenate. This is how strings work.

Let’s move on to the next example where we will talk about lists.. In a list, we can have different types of data as part of one list. We have one "Python programming" string, number, real number, string, and real number. Print and see what the output is. And we can have all elements of the same type? Yes. Print and see what the output is.

How to print the first element? It is by using index0. 0 is the first element, we will get first element, you can print first one. If you want to print elements starting from the 3rd element

index is 2. And print up to last. To print the list two times, the tinylist variable with 5 elements will be repeated in the same order 2 times.

We can also concatenate the list to the first: it would be list+tinylist.

Run the program and see the output. If you have made changes on the fly as interpreters allow us. Save the file and run it. In the output, the first element is "Python Programming" and second element is 100, third is 92.75.

Print elements starting from the third element means, printing 92.75, “john”, and 82.3. Tinylist two times, yes it is here. You can see the both lists are concatenated.

Let’s continue looking at two more items. All I am going to do is, keep introducing various types available in Python. Let's look at this. A set is an unordered collection of items without duplicates. Here the duplicates, orange and apple. They will take only one element “apple” and one element “banana”. Then we can print the basket with apple, orange, pear, and banana.

We can also, check if a specific element in the object basket is there or not. It will check and response a TRUE value or FALSE value. Crabgrass is not in the basket, so this should result in a false value. This is how the set operations are performed. You can see more details in a chapter specific class. Now we are going to move onto Dictionaries.

The dictionary is something where you have key and value pairs. Here IBM has the value of 658.42, and INTEL has the value of 432.80. If you print you will get further details. You can also

append the third key with CISCO with this value of 217.55. You can print this and find out that there are three elements here. If you want to print a specific element in the dictionary all you want to do is, dic['IBM'] or any other value. If you want to delete you can use the key, and say delete. To check if something is in the dictionary what we can use is in IBM. To check if something is not in the dictionary, we can use CISCO.

Run the program and see the output. Look at this, first we have a range of values printed. In this set banana, pear, orange, and apple are four elements. Orange, apple comes only once. Orange is in the basket. So, this becomes true. Crabgrass is not in basket, so, it is false, orange is true. Then you have a list of elements, IBM and INTEL the key in the dictionary has initially been added. CISCO is added, when you printed, and they come in. Trying to access element IBM, we go here. Delete 'IBM' and re assigned. IBM in the dictionary is True.CISCO is not in the dic, so it is false. I hope you have some understanding of set and dictionary variables.

Part 5: Programming Features in Python

Slide 24

Having seen the identifiers and operators, including assignment operators, it is important that we also understand the other language constructs. One of them is about decision making and the other is about iterations, i.e., repetitively performing the same action.

Let me explain how decision making statements are used in Python. An if statement is the basic decision making statement

in Python. It can also have an else part to it. If and else statements can perform simple tests. The bodies of the if and else clauses should be denoted by indentation. Remember we talked about strict enforcement of indentation by Python. The else clause is optional. For Boolean expressions, you should use or, and, and not keywords.

It's important to note that Python does not have a special switch or case statement for testing values. Then you can ask me, what do we do for multiple-test cases? In Python, you can use elif statements for handling multiple cases.

Let's now look at repeating the same task, or set, of statements, subject to a certain condition or conditions. Python has a rich set of features for iteration. However, the most common form of iteration is to simply loop over all the members of a sequence such as a string, list, dictionary, file, or tuple. The most widely used looping constructs are for and while statements. While repeats a statement or group of statements when the condition is true. First, it checks for the condition in the loop and then it executes. For executes a sequence of statements multiple times until the condition fails. In the case of nested loops, you can use one or more loops inside any another while, for, or do...while loop.

We should talk about how you can handle exceptions. You don't want your program to bomb, so Python provides an exception handling feature. An exception/error can be caught and handled. You can also use traceback messages. A traceback message indicates the type of error that occurred, along with its

location. We will see more about error handling in another module.

Part 6: DEMO: Conditional Statements

Let me explain about conditional statements in Python. I will give you two variants. One is IF statement, and the other one is IF and ELSE variant. Then I also explain While statement followed by introduction for loop.

In this case I assign a variable variable with 100. Checking the value is 0 or null. If it is null print nothing. If there is the value, I am expecting it will print "A true expression value".

Then, I am checking an element var while 0 here. Checking if value 0 or null. In this case, I will be printing "Good bye".

Let us look at the while loop, this will print 5 times, starting from 0, 1, 2, 3 and 4. Final statement "Good Bye!" is printed after coming out of the loop.

Let also look into for loop. It is similar to any other programming languages. You can use the range variable here. The values starting from 0 to 4.

In the real world, you will write much lengthier and bigger code. Here I used simple code to explain the concepts. Now Run the program and see the output.

The variable var is 100. So, the true expression value statement is printed. In this case the var value is 0. So, it will say straightaway "Good Bye". You have a while statement here.

It prints the five elements from 0 to 4. Then look at the for loop, again it prints from 0 to 4. This is the loop working in Python. Hope you have a good understanding of a conditional statements.

Let's get back to the lesson.

[end demo]

Part 7: Modules in Python

Slide 25

If your programs are big in size, you should break them into multiple files for better maintenance. These are called modules. A module organizes your Python code logically. Simply put, a module is a file consisting of Python code, definitions, and statements. A module can define functions, classes, and variables. A module may also contain runnable code.

Create a module and put all the relevant statements and definitions into the file. To use your module in other programs, you can use the import statement. Any Python source file can be used as a module by simply executing an import statement in any other source file. After importing, to access the contents of the namespace, just use the name of the module as a prefix directory (dir) function is used to list out the contents of a module. Experimentations are done interactively by using a dir function.

A package is a hierarchical file directory structure that defines a single Python application environment. It consists of modules, subpackages, sub-subpackages, and so on.

Slide 26

We write programs to get an outcome. It could be to a screen or to a file. The simplest way to produce an output is using the print statement where you can pass zero or more expressions separated by commas. This print function converts the expressions and produces a standard output format in a string. Python uses Input and Output functions, such as Open, Close, Read, Write, etc. The open function returns a new file object. By invoking methods on this object, we can perform various file operations which we need. The readline method reads a single line of input, including the termination of newline. The empty string is returned at the end of the file. We can get an output of a program by selecting "go to a file." The write function is used to write raw data. The close function is used to close the entire function.

The same techniques can be applied to the standard output and input streams of the interpreter. They are sys.stdin and sys.stdout. If we want to read user input interactively, we can read from the file sys.stdin. If we want to write data to the screen, we can write to sys.stdout. By using the print statement we can use the same file to output data. These are the input and output functions of Python. Python uses a data interchange format called JavaScript Object Notation, or JSON.

Slide 27

A function is a block of organized, reusable code which performs a single related action. Functions offer better modularity and an excellent reusability of codes. You can use the def statement to create a function. As we already said, Python gives you many

built-in functions like print, but you can also create your own functions. These functions are called user-defined functions. Some rules are there for defining functions. Blocks inside a function generally begin with the keyword def and then a function name in parentheses. Any input parameters or arguments you want to pass can be written within these parentheses. You can also define parameters inside these parentheses. The code block within every function starts with a colon and is indented. A return statement without any arguments are considered empty. As we mentioned earlier, you can use a tuple to return multiple values from a function.

Slide 28

We're going to move on to talking about generators. If you use a yield statement, it can generate an entire sequence of results instead of returning a single value from each function. Any function that uses yield is called a generator. Calling a generator function creates an object.

To modify the value of a global variable from inside a function, you can use a global statement.

Anonymous functions can be created using the lambda keyword. Functions are said to be anonymous if they are not declared in the standard manner, that is, without using the def keyword. Lambda is not limited, so it can use any number of arguments, but returns only one value in the form of an expression. They cannot contain commands or multiple expressions. Next comes the coroutines. Generally, functions operate on a single set of input arguments. However, a function can also be written to operate as a task that processes a sequence of inputs sent to

it. This type of function is called a coroutine. Coroutine is created by using the yield statement as an expression: (yield).

Slide 29

Python has Python DB-API to enable access to databases. Python supports a wide range of database servers, such as GadFly, mSQL, MySQL, PostgreSQL, Microsoft SQL Server 2000, Informix, Interbase, Oracle, and Sybase.

In Python, you have a separate database API module for the corresponding database. For example, if you require access to an mSQL database as well as a MySQL database, you must have both the mSQL and the MySQL database APIs.

Databases that support Python are dbm (an interface based on ndbm), GDBM (GNU's reinterpretation of dbm), and sqlite3. Sqlite3 is a database which provides a lightweight disk-based relational database. Relational databases are the most widely used type of database. The DB-API is a common interface for relational databases. The current version of the specification is version 2.0. Interfaces to disk-based hashes such as dbm and GDBM are also included with standard Python. Python has a generic DB-API with ODBC support. Some of them are mxODBC, pyodbc, ceODBC, odbtpapi, PyPyODBC, and the win32::odbc module. Python DB-API is independent of any database engine, which enables us to write Python scripts to access any database engine. It supports databases such as MySql, Oracle, and Microsoft SQL Server.

How do you access a database? Initially, you have to import the API module, then establish a connection with the database. Then

write SQL statements and create stored procedures for your program. Once done, you can destroy the connection.

To summarize, Python supports access to any database and you have APIs to manage.

Slide 30

The Common Gateway Interface, or CGI, is a gateway with standards that tell us how information is exchanged between the web server and a custom script. CGI is also an external gateway program to interface with information servers such as HTTP servers. The current version is CGI/1.1. The upcoming version is CGI/1.2.

A regular expression is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax. Regular expressions are widely used in the Unix world.

The match function matches RE (regular expression) pattern to a string with optional flags. The search function searches for the initial occurrence of the RE pattern within a string with optional flags. To search and replace, the most important method that regular expressions use is sub. This method replaces all occurrences of the RE pattern with repl until the substituting reaches a maximum. This method should return a modified string.

Part 8: Extensions & OOPs in Python

Slide 31

Any code that you write using a compiled language, like C, C++, or Java, can be integrated or imported into another Python

script. This code is called an “extension.” First look at a Python extension; you'll be grouping your code into four parts. The first part is the header file Python.h. The second is those C functions which you want to expose as the interface from your module. Third is a table mapping the names of your functions to C functions inside the extension module. Last is the initialization function.

Here’s an overview about object-oriented programming. Programmers are recommended to use object-oriented programming concepts in their Python scripts as they are easier to use when creating and handling classes and objects in programs.

All values that are used in a program are called objects. An object consists of internal data and methods that perform various kinds of operations involving that data. We have already used objects and methods while working with built-in types such as strings and lists.

A user-defined prototype that accesses attributes which characterize any object in a block is called a class. What are these attributes? Attributes are nothing but variables used in class, method, and instance variables that are defined using dot notation. The class or instance variable that holds values of class and its objects is called a data member. Instance variables are defined inside a method and can be accessed only inside that class. The class defines a simple stack with push(), pop(), and length() operations. We’ll see more about these operations in following modules.

Inheritance is a property used to transfer or modify the characteristics of the original class to a new class. In other

words, new classes are derived from the base class. The original class is called the base class. The new class that is created from the base class is a derived class or a subclass. It inherits the objects that are defined by its base classes. A derived class may also redefine any of these attributes and add new attributes of its own. Inheritance can be used as a list of base-class names separated by commas in the class statement.

Function overloading is the assignment of more than one behavior to a particular function. The operation performed varies by the types of arguments involved. The assignment of more than one function to a particular operator is operator overloading. Classes can overload all Python expression operators. Operator overloading lets classes intercept normal Python operations.

Polymorphism is a method signifying that the meaning of an operation depends on the type of the objects or attributes being operated on. For example: (x+). Here the plus sign operates based on the objects.

Part 9: Conclusion

This is the end of our overview of Python. If we look back over what we've discussed, the features seem overwhelming. But you should now have a basic understanding of Python, especially the data types, how they can be operated upon, decision making, and repeat language constructs. In addition, you now have an idea about functions and how you may modularize a Python program. Python provides an interface with databases and provides a very good CGI to develop Internet programs. To conclude, Python supports object orientation and the processing of regular expressions. Python is simple, portable, and easy to develop and

use. We am sure you will enjoy learning this language and use it extensively to develop applications to support your business.

Thank you very much for your patient listening. For further information on this and related topics, reach out to us and we will be more than happy to assist you.