



LOVELY
PROFESSIONAL
UNIVERSITY

SIX WEEKS SUMMER TRAINING REPORT
On

Python for everybody

Submitted by
Vishal Gupta

Registration No: 11704487

Programme Name: Btech. CSE (3rd Year)

School of Computer Science & Engineering

Lovely Professional University, Phagwara

(June-July,2021)

Vishal gupta

DECLARATION

hereby declare that I have completed my six weeks summer training at **coursera** platform from May 01,2020 to July 02,2020 under the guidance of Charles Russell Severance. I declare that I have worked full dedication during their 8 weeks of training and my learning outcomes fulfil the requirements of training for the award of degree of **B.tech. CSE,Lovely Professional University, Phagwara.**

Name: Vishal GUPTA

Registration no: 11704487

Date: 26 September ,2021


ACKNOWLEDGEMENT

I would like to express my gratitude towards my university as well as **coursera** for providing me the golden opportunity to do this wonderful summer training *regarding python for everybody*, which also helped me in doing a lot of homework and learning. As a result, I came to know about so many new things. So, I am really thankful to them.

Moreover, I would like to thank my friends who helped me a lot whenever I got stuck in some problem related to my course. I am really thankful to have such a good support of them as they always have my back whenever I need.

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

Summer Training Certificate by *coursera*



5 Courses


Programming for Everybody
(Getting Started with Python)

Python Data Structures

Using Python to Access Web Data

Using Databases with Python

Capstone: Retrieving, Processing, and Visualizing Data with Python




Jul 15, 2021

VISHAL GUPTA

has successfully completed the online, non-credit Specialization

Python for Everybody

This Specialization builds on the success of the Python for Everybody course and will introduce fundamental programming concepts including data structures, networked application program interfaces, and databases, using the Python programming language. In the Capstone Project, you'll use the technologies learned throughout the Specialization to design and create your own applications for data retrieval, processing, and visualization.


Charles Severance
Clinical Associate
Professor, School of
Information
University of Michigan

The online specialization named in this certificate may draw on material from courses taught on-campus, but the included courses are not equivalent to on-campus courses. Participation in this online specialization does not constitute enrollment at this university. This certificate does not confer a University grade, course credit or degree, and it does not verify the identity of the learner.

Verify this certificate at:
coursera.org/verify/specialization/QEXFUP6VPSL2

<https://coursera.org/share/8296d46bf356fa9a70a96dbb909c2ac9>



Jul 15, 2021

VISHAL GUPTA

has successfully completed

Programming for Everybody (Getting Started with Python)

an online non-credit course authorized by University of Michigan and offered through Coursera

A handwritten signature in black ink, appearing to read 'Charles', followed by a horizontal line.

Charles Severance
Clinical Professor, School of Information
University of Michigan

COURSE
CERTIFICATE



Verify at coursera.org/verify/AYLG4EG63JJP

Coursera has confirmed the identity of this individual and their participation in the course.

<https://coursera.org/share/5edf9f11e1d13fe816a2114c86292dca>



Jul 15, 2021

VISHAL GUPTA

has successfully completed

Using Python to Access Web Data

an online non-credit course authorized by University of Michigan and offered through Coursera

A handwritten signature in black ink, appearing to read 'Charles', followed by a horizontal line.

Charles Severance
Clinical Professor, School of Information
University of Michigan

COURSE
CERTIFICATE



Verify at coursera.org/verify/S7RPLT35UEC5

Coursera has confirmed the identity of this individual and their participation in the course.



<https://coursera.org/share/5edf9f11e1d13fe816a2114c86292dca>



Jul 15, 2021

VISHAL GUPTA

has successfully completed

Python Data Structures

an online non-credit course authorized by University of Michigan and offered through Coursera

A handwritten signature in black ink, appearing to read 'Charles', followed by a horizontal line.

Charles Severance
Clinical Professor, School of Information
University of Michigan

COURSE
CERTIFICATE



Verify at coursera.org/verify/ASQGSJCUWSZR
Coursera has confirmed the identity of this individual and their
participation in the course.

<https://coursera.org/share/e23974b73a064d754fb311cc2628e412>



<https://coursera.org/share/aff1731afe04a1489a9bd2048370480a>



<https://coursera.org/share/3c9963aa69f2bf9e1638e58c31c31cf2f0>

CONTENTS

<i>S.No.</i>	<i>Title</i>	<i>Page No.</i>
1	introduction	12
2	Technology learnt	13
3	Reason for choosing Python for Everybody Specialization	44
4	Learning outcome	45
5	bibliography	46

INTRODUCTION

The **Python for Everybody specialization** is considered one of the famous online courses out there to learn 1 million python with over enrolment which is awesome This Specialization builds on the success of the Python for Everybody course and I have learn fundamental programming concepts including data structures, networked application program interfaces, and databases, using the Python programming language. In the Capstone Project, I will use the technologies learned throughout the Specialization to design and create our own applications for data retrieval, processing, and visualization.

Python for everybody specialization course having five courses

- 1.Programming for Everybody (Getting Started with Python)
- 2.Python Data Structures
- 3.Using Python to Access Web Data
- 4.Using Databases with Python
- 5.Capstone: Retrieving. Processing, and Visualizing Data with Python

TECHNOLOGY LEARNT

It had 5 courses which was further divided into chapters and then topics so during my whole course I learned the following:

INTRODUCTION TO Python for everybody specialization

= > Programming for Everybody (Getting Started with Python)

1. Python
2. Object Oriented programming
3. History of Python
4. Behind the Scenes of python
5. Installing Python and Setup the path Variable

= > Python Data Structures

1. Tuple
2. List
3. Dictionary
4. Sets

= > Using Python to Access Web Data

1. Regular Expression
2. Extracting data
3. Hypertext Transfer Protocol (HTTP)
4. Networked Technology
5. Unicode Characters and Strings
6. Parsing Web Pages
7. extensible Markup Language (XML)

8. JavaScript Object Notation (JSON)

9. Using Application Programming Interfaces

=> Using Databases with Python

=> Capstone: Retrieving, Processing, and Visualizing Data with Python

1. Programming for Everybody (Getting Started with Python)

o Python

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and

comprehensive standard library. Python interpreters are available for installation on many operating systems, allowing Python code execution on a wide variety of systems. Python is commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it's relatively easy to learn, Python has been adopted by many non-programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances.

o Object Oriented Programming Language

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods. A distinguishing feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). In OOP programming, computer programs are designed by making them out of objects that interact with one another. There is significant diversity in object oriented programming, but most popular languages are class-based,

meaning that objects are instances of classes, which typically also determines their type.

o History

Python was conceived in the late 1980s, and its implementation was started in December 1989 by Guido van Rossum at CWI in the Netherlands as a successor to the ABC language (itself inspired by SETL) capable of exception handling and interfacing with the Amoeba operating system. Van Rossum is Python's principal author, and his continuing central role in deciding the direction of Python is reflected in the title given to him by the Python community, benevolent dictator for life (BDFL). In the late 1980s, history was about to be written. It was that time when working on Python started. Soon after that, Guido Van Rossum began doing its application-based work in December of 1989 by at Centrum Wickenden

& Informatica (CWI) which is situated in Netherland. It was started firstly as a hobby project because he was looking for an interesting project to keep him occupied during Christmas. The programming language which Python is said to have succeeded is ABC Programming Language, which had the interfacing with the Amoeba Operating System and had the feature of exception handling. He had already helped to create ABC earlier in his career and he had seen some issues with ABC but liked most of the features. After that what he did as really very clever. He had taken the syntax of ABC, and some of its good features. It came with a lot of complaints too, so he fixed those issues completely and had created a good scripting language which had removed all the flaws.

Behind The Scenes of Python

About the origin of Python, Van Rossum wrote in 1996:

Over six years ago, in December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office... would be closed, but I had a home Computer, and not much else on me hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers.

I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus). While I am trying to be as

precise as possible, I chose to use English rather than formal specifications for everything except syntax and lexical analysis. This should make the document more understandable to the average reader but will leave room for ambiguities.

Consequently, if you were coming from Mars and tried to re-implement Python from this document alone, you might have to guess things and in fact you would

probably end up implementing quite a different language. On the other hand, if

you are using Python and wonder what the precise rules about a particular area of

the language are, you should definitely be able to find them here. So Python is not

defined by its language reference only. It would be also wrong to say that Python

is defined by its reference implementation, CPython, since there are some implementation details that are not a part of the language. The garbage collector

that relies on a reference counting is one example. Since there is no single source

of truth, we may say that Python is defined partly by the Python Language Reference and partly by its main implementation, Python.

o Installing Python

Once you have downloaded the Python MSI, simply navigate to the download location on your computer, double clicking the file and pressing Run when the dialog box pops up.

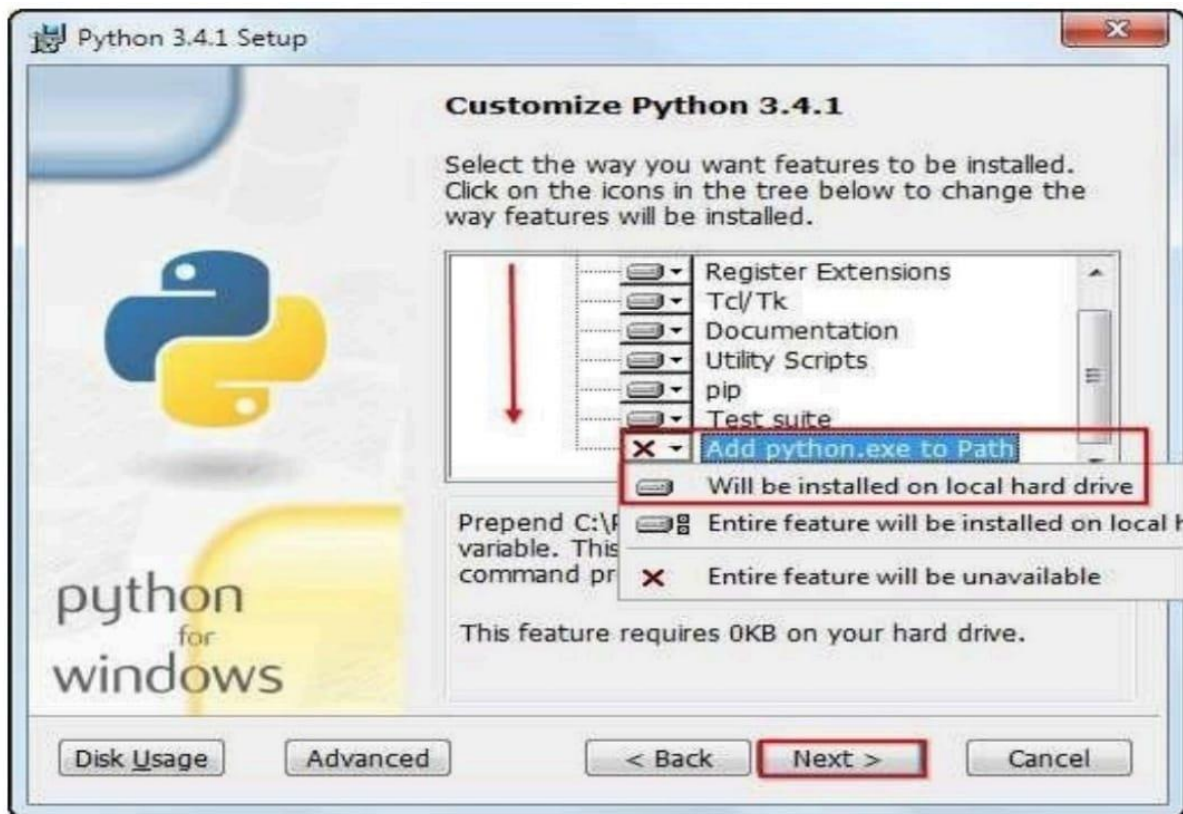


If you are the only person who uses your computer, simply leave the "Install for all users" option selected. If you have multiple accounts on your PC and don't want to install it across all accounts, select the "Install just for me" option then

press "Next."



if you want to change the install location, feel free to do so; however, it is best to leave it as is and simply select next, Otherwise... Scroll down in the window and find the "Add Python.exe to Path" and click on the small red "x." Choose the "Will be installed on local hard drive" option then press "Next."



Now that you have completed the installation process, click on "Finish."

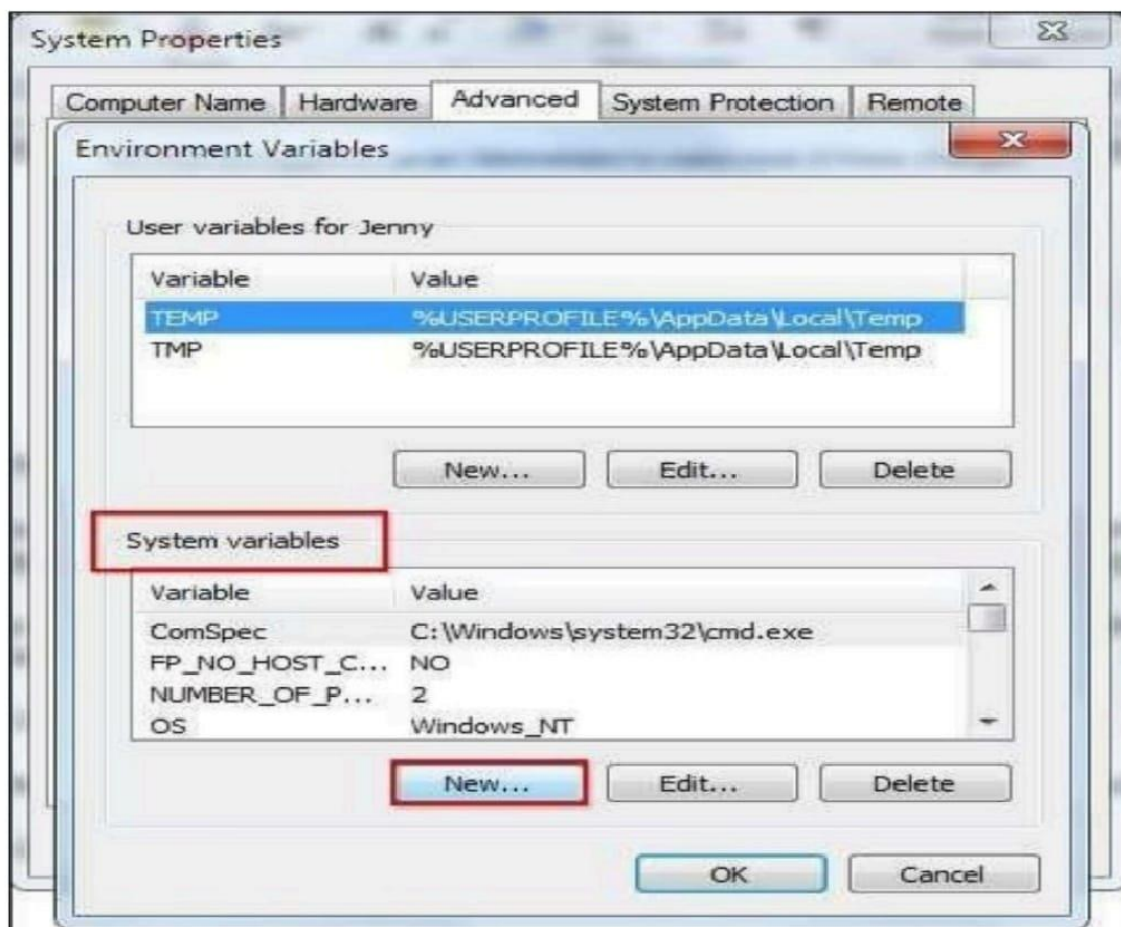


Setup the path Variable

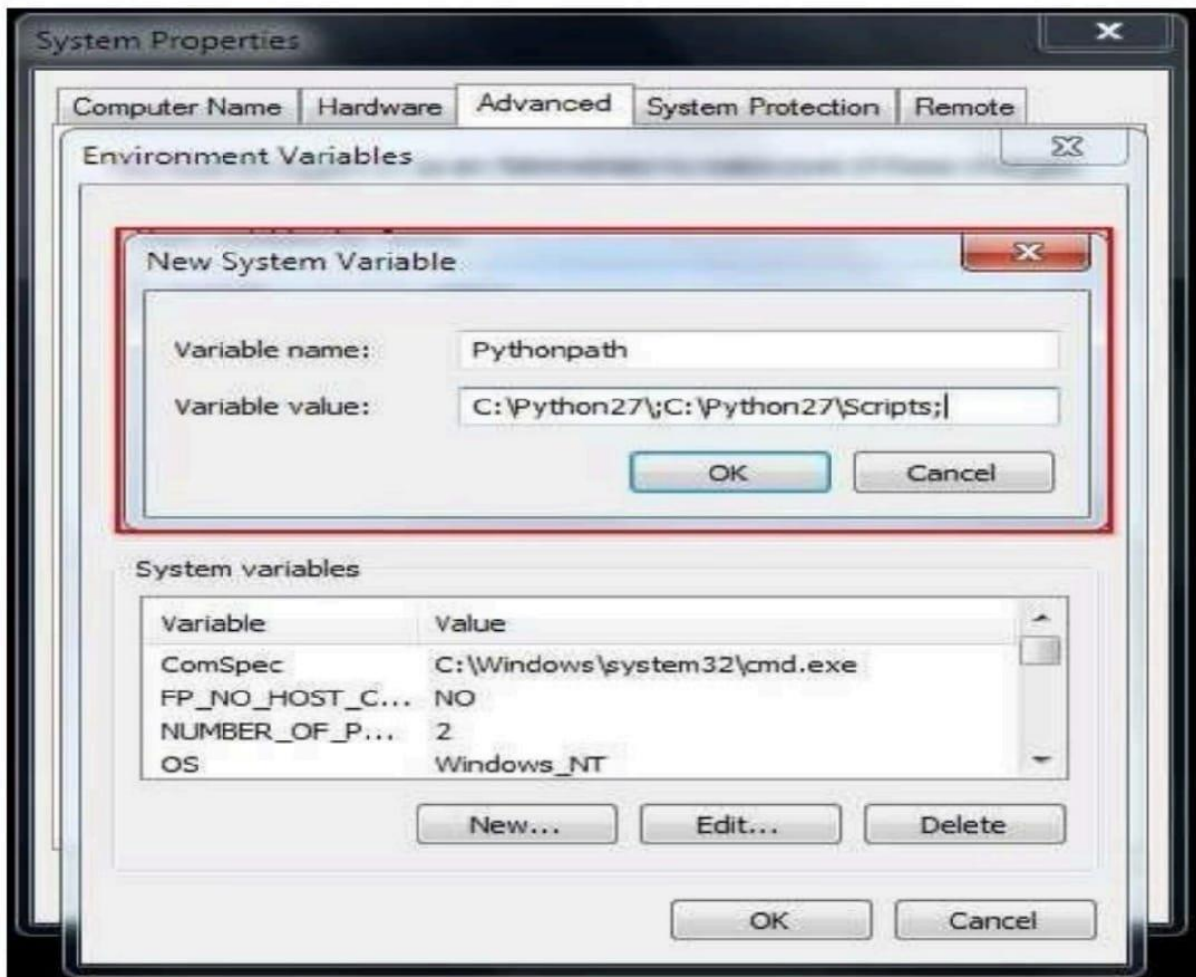
Begin by opening the start menu and typing in "environment" and select the option called

"Edit the system environment variables."

When the "System Properties" window appears, click on "Environment Variables..." Once you have the "Environment Variables" window open, direct your focus to the bottom half. You will notice that it controls all the "System Variables" rather than just this associated with your user. Click on "New..." to create a new variable for Python.



Simply enter a name for your Path and the code shown below. For the purposes of this example we have installed Python 2.7.3, so we will call the path: "Python path."
The string that you will need to enter is: "C:\Python 27\; C:\Python27\Scripts"



Running The Python IDE

Now that we have successfully completed the installation process and added us

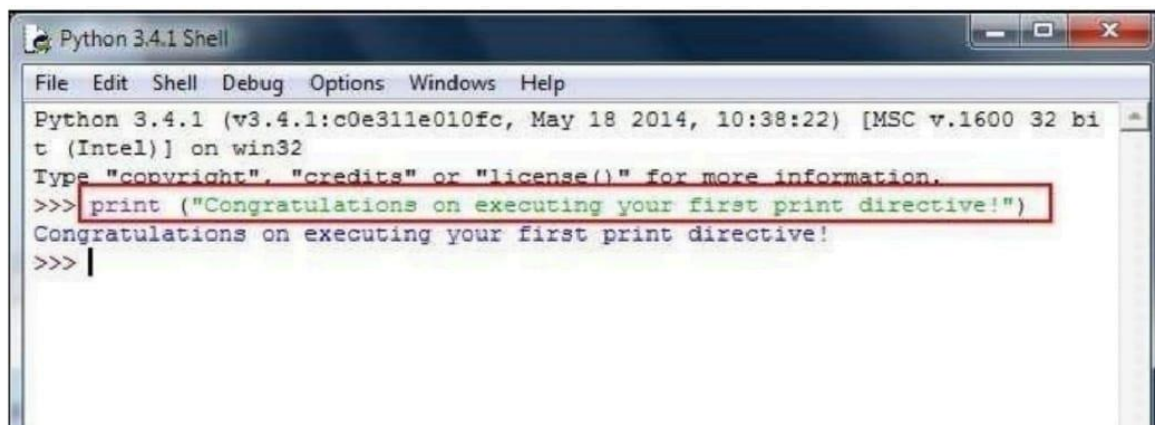
"Environment Variable," you are ready to create your first basic Python script. Let's begin by opening Python's GUI by pressing "Start" and typing "Python" and selecting the "IDLE (Python GUI)."



Once the GUI is open, we will begin by using the simplest directive possible. This is the "print" directive which simply prints whatever you tell it to, into a new line.

Start by typing a print directive like the one shown in the image below or copy and paste this text then press "Enter": `print ("Congratulations on executing your first print directive!")`

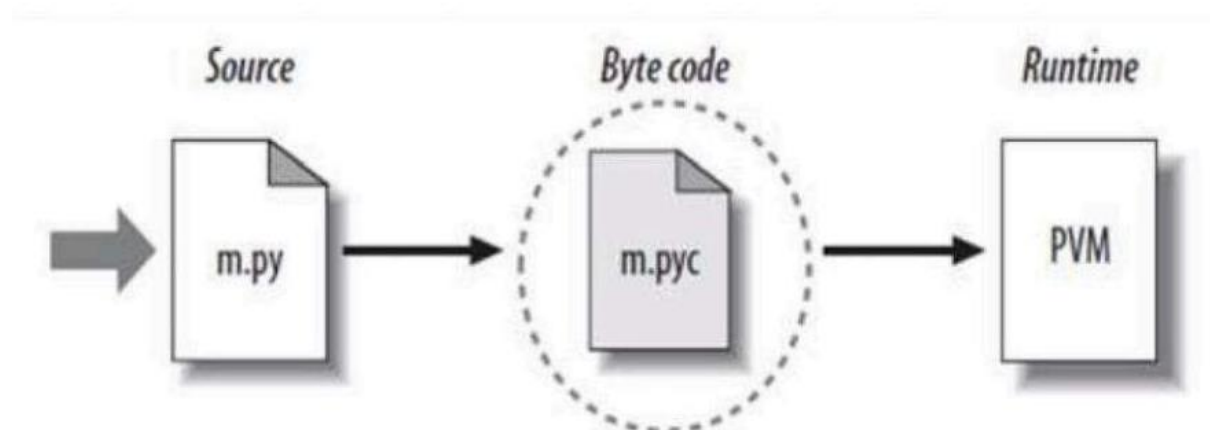
Python



```
Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bi
t (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print ("Congratulations on executing your first print directive!")
Congratulations on executing your first print directive!
>>> |
```

Python Code Execution

Python's traditional runtime execution model: source code you type is translated to byte code, which is then run by the Python Virtual Machine. Your code is automatically compiled, but then it is interpreted.



Source code extension is .py Byte code extension is .pyc (compiled python code)

o Data Type

(This is called dynamic typing). Data types determine whether an object can do something, or whether it just would not make sense. Other programming languages often determine whether an operation makes sense for an object by making sure the object can never be stored somewhere where the operation will

be performed on the object (this type of system is called static typing). Python does

not do that. Instead, it stores the type of an object with the object, and checks when

the operation is performed whether that operation makes sense for that object

Python has many native data types. Here are the important ones:

Booleans are either True or False.

Numbers can be integers (1 and 2), floats (1.1 and 1.2), fractions (1/2 and 2/3), or even complex numbers.

Strings are sequences of Unicode characters, e.g., an HTML document.

Bytes and byte arrays, e.g., a JPEG image file.

Lists are ordered sequences of values.

Tuples are ordered, immutable sequences of values.

Sets are unordered bags of values.

o Variable

Variables are nothing but reserved memory locations to store values. This means

that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides

what can be stored in the reserved memory. Therefore, by assigning different data

types to variables, you can store integers, decimals or characters in these variables.

Ex:

```
counter = 100 # An integer
```

```
assignment miles = 1000.0 # A floating
```

```
point name = "John" # A string
```

String

In programming terms, we usually call text a string. When you think of a string as a collection of letters, the term makes sense. All the letters, numbers, and symbols in this book could be a string. For that matter, your name could be a string, and so could your address.

Creating Strings

In Python, we create a string by putting quotes around text. For example, we could take us

```
# "hello"+"world" "helloworld"
```

o Python Operator

Arithmetic Operator

Operator	Meaning	Example
+	Add two operands or unary plus	x + y +2
-	Subtract right operand from the left or unary minus	x - y -2
*	Multiply two operands	x * y
/	Divide left operand by the right one (always results into float)	x / y

%	Modulus - remainder of the division of left operand by the right	$x \% y$ (remainder of x/y)
//	Floor division - division that results into whole number adjusted to the left in the number line	$x // y$
**	Exponent - left operand raised to the power of right	$x ** y$ (x to the power y)

Comparison Operator

>	Greater than - True if left operand is greater than the right	$x > y$
<	Less than - True if left operand is less than the right	$x < y$
==	Equal to - True if both operands are equal	$x == y$
!=	Not equal to - True if operands are not equal	$x != y$
>=	Greater than or equal to - True if left operand is greater than or equal to the right	$x >= y$
<=	Less than or equal to - True if left operand is less than or equal to the right	$x <= y$

2. Python Data Structures

o Tuple

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are the tuples cannot be changed unlike lists and tuples use parentheses.

Accessing Values in Tuples:

To access values in tuple, use the square brackets for slicing along with the index

or indices to obtain value available at that index. For example - tup1 =

('physics',

'chemistry', 1997, 2000); tup2 = (1, 2, 3, 4, 5, 6, 7); print "tup1[0]: ", tup1[0]

print

"tup2[1:5]: ", tup2[1:5]

When the above code is executed, it produces the following result-tup1[0]:

physics

tup2[1:5]: [2, 3, 4, 5]

Basic Tuples Operations

Tuples respond to the + and operators much like strings; they mean concatenation

and repetition here too, except that the result is a new tuple, not a string. In fact,

tuples respond to all of the general sequence operations we used on strings in the

prior chapter –

Python Expression	Results	Description
<code>len((1, 2, 3))</code>	3	Length
<code>(1, 2, 3) + (4, 5, 6)</code>	(1, 2, 3, 4, 5, 6)	Concatenation
<code>('Hi!') * 4</code>	('Hi!', 'Hi!', 'Hi!', 'Hi!')	Repetition
<code>3 in (1, 2, 3)</code>	True	Membership
<code>for x in (1, 2, 3): print x,</code>	1 2 3	Iteration

Built-in Tuple Functions

Python includes the following tuple functions

SN	Function with Description
1	<code>cmp(tuple1, tuple2)</code> Compares elements of both tuples.
2	<code>len(tuple)</code> Gives the total length of the tuple.
3	<code>max(tuple)</code> Returns item from the tuple with max value.
4	<code>min(tuple)</code> Returns item from the tuple with min value.
5	<code>tuple(seq)</code> Converts a list into tuple.

o List

The list is a most versatile datatype available in Python which can be written as a list of commas separated values (items) between squares

brackets. Important thing about a list

is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values

between square brackets. For example - list1 = ['physics', 'chemistry', 1997, 2000]; list2 = [1, 2, 3, 4, 5]; list3 = ["a", "b", "c", "d"].

Like string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

Accessing Values in Lists:

To access values in lists, use the square brackets for slicing along with the index or indices to obtain

value available at that index. For example - list1 = ['physics', 'chemistry', 1997, 2000]; list2 = [1, 2, 3, 4, 5, 6, 7]; print "list1[0]: ", list1 [0] print "list2[1:5]", list2[1:5]

Output: list1[0]: physics

list2[1:5]: [2, 3, 4, 5]

Update: list= ['physics', 'chemistry', 1997, 2000]; print

"Value available at index 2: " print list [2] list [2] = 2001; print

"New value available at index 2: " print list [2]

Output: Value available at index 2: 1997

New value available at index 2: 2001

Delete: list1= ['physics', 'chemistry', 1997, 2000]; print list1

del list1[2]; print "After deleting value at index 2: " print list1

['physics', 'chemistry', 1997, 2000]

Output: After deleting value at index 2:

['physics', 'chemistry", 2000]

Basic List Operation

Python Expression	Results	Description
<code>len([1, 2, 3])</code>	3	Length
<code>[1, 2, 3] + [4, 5, 6]</code>	<code>[1, 2, 3, 4, 5, 6]</code>	Concatenation
<code>['Hi!'] * 4</code>	<code>['Hi!', 'Hi!', 'Hi!', 'Hi!']</code>	Repetition
<code>3 in [1, 2, 3]</code>	True	Membership
<code>for x in [1, 2, 3]: print x,</code>	1 2 3	Iteration

Built-in List Functions & Methods:

SN	Function with Description
1	<code>cmp(list1, list2)</code> Compares elements of both lists.
2	<code>len(list)</code> Gives the total length of the list.
3	<code>max(list)</code> Returns item from the list with max value.
4	<code>min(list)</code> Returns item from the list with min value.
5	<code>list(seq)</code> Converts a tuple into list.

Python includes following list methods

SN	Methods with Description
1	<u>list.append(obj)</u> Appends object obj to list
2	<u>list.count(obj)</u> Returns count of how many times obj occurs in list
3	<u>list.extend(seq)</u> Appends the contents of seq to list
4	<u>list.index(obj)</u> Returns the lowest index in list that obj appears
5	<u>list.insert(index, obj)</u> Inserts object obj into list at offset index
6	<u>list.pop(obj=list[-1])</u> Removes and returns last object or obj from list

o Dictionary

Dictionary in Python is an unordered collection of data values, used to store data

values like a map, which, unlike other Data Types that hold only a single value as an element, Dictionary holds key: value pair. Key-value is provided in the dictionary to make it more optimized.

Creating a Dictionary

In Python, a Dictionary can be created by placing a sequence of elements within

curly {} braces, separated by 'comma'. Dictionary holds a pair of values, one being the Key and the other corresponding pair element being its Key: value. Values in a dictionary can be of any data type and can be duplicated, whereas keys can't be repeated and must be immutable.

Creating a Dictionary

With Integer Keys

```
Dict = {1: 'Geeks', 2: 'For', 3: 'Geeks'}
```

```
print ("\nDictionary with the use of Integer Keys: ")
```

```
print (Dict)
```

```
# Creating a Dictionary
# With Mixed keys
Dict = {'Name': 'Geeks', 1: [1, 2, 3, 4]}
print ("\nDictionary with the use of Mixed Keys: ")
print (Dict)
```

o Sets

A Set is an unordered collection data type that is iterable, mutable and has no duplicate elements. Python's set class represents the mathematical notion of a set.

The major advantage of using a set, as opposed to a list, is that it has a highly optimized method for checking whether a specific element is contained in the set.

This is based on a data structure known as a hash table. Since sets are unordered,

we cannot access items using indexes like we do in lists.

```
# Python program to
# Demonstrate sets
# Same as {"a", "b", "c"}
myset = set (["a", "b", "c"])
print(myset)
# Adding element to the set
myset.add("d")
print(myset)
```

Using python to access web data

o Regular Expression

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 held in a pattern.

Regular expressions are widely used in UNIX world.

The Python module re provides full support for Perl-like regular expressions in Python. The re module raises the exception re.error if an error occurs while compiling or using a regular expression

We would cover two important functions, which would be used to handle regular expressions. But a small thing first: There are various characters, which would have special meaning when they are used in regular expression. To avoid any confusion while dealing with regular expressions, we would use Raw Strings as r expression'.

o Hypertext Transfer Protocol (HTTP)

The Hypertext Transfer Protocol is an application protocol for distributed, collaborative, hypermedia information systems that allows users to communicate data on the World Wide Web.

The Hypertext Transfer Protocol is an application protocol for distributed, collaborative, hypermedia information systems that allows users to communicate data on the World Wide Web.

HTTP was invented alongside HTML to create the first interactive, text-based web browser: the original World Wide Web. Today, the protocol remains one of the primary means of using the Internet.

o Networked Technology

Python provides two levels of access to network services. At a low level, you can access the basic socket support in the underlying operating system, which allows you to implement clients and servers for both connection-oriented and connectionless protocols.

Python also has libraries that provide higher-level access to specific application level

network protocols, such as FTP, HTTP, and so on.

This chapter gives you understanding on most famous concept in Networking - Socket Programming.

Sockets are the endpoints of a bidirectional communications channel. Sockets may communicate within a process, between processes on the same machine, or between processes on different continents.

Sockets may be implemented over a number of different channel types: Unix domain sockets, TCP, UDP, and so on. The socket library provides specific classes for handling the common transports as well as a generic interface for handling the rest.

o Unicode Characters and Strings

NLP models often handle different languages with different character sets. Unicode is a standard encoding system that is used to represent characters from almost all languages. Every Unicode character is encoded using a unique integer code point between 0 and 0x10FFFF. A Unicode string is a sequence of zero or more code points.

This tutorial shows how to represent Unicode strings in TensorFlow and manipulate them using Unicode equivalents of standard string ops. It separates Unicode strings into tokens based on script detection.

o Parsing Web Pages

Parsing means analysing and converting a program into an internal format that a runtime environment can actually run, for example the JavaScript engine inside browsers.

The browser parse HTML into a DOM tree. HTML parsing involves tokenization and tree construction. HTML tokens include start and end tags, as well as attribute names and values. If the document is well-formed, parsing it is straightforward and faster. The parser parses tokenized input into the document, building up the document tree.

When the HTML parser finds non-blocking resources, such as an image, the browser will request those resources and continue parsing. Parsing can continue

when a CSS file is encountered, but `<script>` tags—particularly those without an `async` or `defer` attribute—blocks rendering and pauses parsing of HTML.

When the browser encounters CSS styles, it parses the text into the CSS Object Model (or CSSOM), a data structure it then uses for styling layouts and painting.

The browser then creates a render tree from both these structures to be able to

paint the content to the screen. JavaScript is also downloaded, parsed, and then execute.

JavaScript parsing is done during compile time or whenever the parser is invoked, such as during a call to a method.

o extensible Markup Language (XML)

Extensible Markup Language (XML) is a universal format, maintained by the W3C, used for representation and transfer of structured data on the web or between different applications.

The language uses a structured representation by allowing users to create custom defined tags according to XML Document Type Definition (DTD) standards. The structure of an XML document can be represented in the form of a tree known as

a Document Object Model (DOM).

o JavaScript Object Notation (JSON)

JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax. It is commonly used for transmitting data in web applications (e.g., sending some data

from the server to the client, so it can be displayed on a web page, or vice versa)

o Using Application Programming Interfaces

When you use an application on your mobile phone, the application connects to

the Internet and sends data to a server. The server then retrieves that data, interprets it, performs the necessary actions and sends it back to your phone.

The application then interprets that data and presents you with the information you

wanted in a readable way. This is what an API is - all of this happens via API.

To explain this better, let us take a familiar example.

Imagine you're sitting at a table in a restaurant with a menu of choices to order from. The kitchen is the part of the "system" that will prepare your order.

What

is missing is the critical link to communicate your order to the kitchen and deliver your food back to your table. That's where the waiter or API comes in. The waiter is the messenger – or API – that takes your request or order and tells the kitchen – the system – what to do. Then the waiter delivers the response back to you; in this case, it is the food.

Here is a real-life API example. You may be familiar with the process of searching flights online. Just like the restaurant, you have a variety of options to choose from, including different cities, departure and return dates, and more. Let

us imagine that you're booking your flight on an airline website. You choose a departure city and date, a return city and date, cabin class, as well as other variables. In order to book your flight, you interact with the airline's website to access their database and see if any seats are available on those dates and what the costs might be.

However, what if you are not using the airline's website—a channel that has direct access to the information? What if you are using an online travel service, such as Kayak or Expedia, which aggregates information from a number of airline databases?

The travel service, in this case, interacts with the airline's API. The API is the interface that, like your helpful waiter, can be asked by that online travel service

to get information from the airline's database to book seats, baggage options, etc.

The API then takes the airline's response to your request and delivers it right back

to the online travel service, which then shows you the most updated, relevant information.

Using Databases with Python

o Designing a Data Model

Data modelling is the process of creating a simple diagram of a complex software system, using text and symbols to represent the way data will flow. The diagram can be used to ensure efficient use of data as a blueprint for the construction of new software or for reengineering a legacy application. Typically, a data model can be thought of as a flowchart that illustrates the relationships among pieces of data. It enables stakeholders to identify errors and make changes before programming code is written. Alternatively, models can be created as part of reverse-engineering efforts that extract models from existing systems, as seen with NoSQL data

o Representing a Data Model in Tables

In relational databases, and flat file databases, a table is a set of data elements (values) using a model of vertical columns (identifiable by name) and horizontal rows, the cell being the unit where a row and column intersect. A table has a specified number of columns, but can have any number of rows

Relational Model (RM) represents the database as a collection of relations. A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.

The table name and column names are helpful to interpret the meaning of values in each row. The data are represented as a set of relations. In the relational model, data are stored as tables. However, the physical storage of the data is independent of the

o Inserting Relational Data

A relational database is a type of database that stores and provides access to data points that are related to one another. ... The columns of the table hold attributes of the data, and each record usually has a value for each attribute, making it easy to establish the relationships among data points.

o Many-to-Many Relationships

A many-to-many relationship occurs when multiple records in a table are associated with multiple records in another table. For example, a many-to-many relationship exists between customers and products: customers can purchase various products, and products can be purchased by many customers. Relational database systems usually don't allow you to implement a direct many-to-many relationships between two tables. Consider the example of keeping track of invoices. If there were many invoices with the same invoice number and one of your customers inquired about that invoice number, you wouldn't know which number they were referring to. This is one reason for assigning a unique value to each invoice.

To avoid this problem, you can break the many-to-many relationship into two one-to-many relationships by using a third table, called a join table. Each record in a join table includes a match field that contains the value of the primary keys of the two tables it joins. (In the join table, these match fields are foreign keys.) These foreign key fields are populated with data as records in the join table are created from either table it joins.

o Using Databases with Python

At a high level web applications store data and present it to users in a useful way.

For example, Google stores data about roads and provides directions to get from one location to another by driving through the Maps application. Driving directions are possible because the data is stored in a structured format.

Databases make structured storage reliable and fast. They also give you a mental framework for how the data should be saved and retrieved instead of having to figure out what to do with the data every time you build a new application. Relational databases store data in a series of tables. Interconnections between the tables are specified as foreign keys. A foreign key is a unique reference from one row in a relational table to another row in a table, which can be the same table but is most commonly a different table

o Identifying Your Data Source

In this week, we want you to identify a data source and make a short discussion forum post about it. We have provided a reading page with a great set of starting points to give you ideas about some data sources that might be interesting to you.

Once you identify and evaluate your data source, we want you to make a short post in the Week 3 Discussion Forum describing the data source and outlining some possible analysis that could be done with the data source. You should direct the post to your fellow students as they might want to do a project using the data source you are presenting.

Making a post is completely optional. If you do not want to make a post, you are still encouraged to review and comment on some of the posts by other students.

In later weeks you will analyse and present your analysis of your data. We are happy for the analysis to be done by either an individual or a group. And even if you make a post on one data source - you are not bound to do a project on that data source. You may see a post from another student that piques your interest and you may choose to join their effort or just do some data analysis on your own using their data source. We want this optional discussion thread track to give you as much flexibility as you need to learn as much as possible from the work of your fellow students

REASON FOR CHOOSING Python for Everybody Specialization

All of the above was part of my training during my summer break I specially choose the Python for Everybody Specialization by coursera for reasons stated below:

1. Being a MOOC enthusiast myself, I want to take as many courses as possible. However, just like most of you, I have limited time, meaning that I really don't want to waste it on low-quality courses.
2. That's why I believe that it's important to look up what other people thought of the course. While everyone is an individual and the opinions might differ from person to person, the consensus can tell a lot about the course.
3. Realizing that it's not that easy to find a well-rounded, non-biased article summarizing the main points of the course, I felt the need to bring it to you. Therefore, after reading the advantages of the Coursera Python course, I'll also show some sides of the course that might be lacking.
4. A specialization is a way more extensive study track that's made from a few courses that follow the same path. Completing a specialization, you'll have a wider array of skills than you would have if you only chose a specific course.
5. Specializations usually take months to finish. Therefore, some people don't really like them. However, if you're serious about learning a subject, specialization is something that you shouldn't think twice about before enrolling.
6. A course, as the name suggests, is a shorter single course on a specific topic. It might cover a wider topic but definitely not in great detail. Courses aren't that demanding of your time, they might take only a few weeks to complete, in some cases only a few hours.

LEARNING OUTCOMES

You will understand the fundamentals of programming instructions and how computers understand the programming languages and know what to do as well

as how the python language works and some basics and general concepts of computer programming.

The course is really intended to be friendly for beginners and start teaching you

from the history of computer and programming and then move to learn python

and its fundamentals

BIBLIOGRAPHY

=> Coursera

=> Geeks for Geeks website

=> Javarevisited

=> Wikipedia

=> w3schools

