CISCO - Virtual Internship Program 2022

An Internship report submitted by

Pratheesh Raj P L - URK21CS1064

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

under the supervision of

Dr. GETZI JEBA LEELIPUSHPAM

DEPARTMENTOF COMPUTER SCIENCE AND ENGINEERING

KARUNYA INSTITUTE OF TECHNOLOGY AND SCIENCES

(Declared as Deemed to be University under Sec-3 of the UGC Act, 1956)
Karunya Nagar, Coimbatore - 641114. INDIA

October 2022

Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)
A CHRISTIAN MINORITY RESIDENTIAL INSTITUTION
AICTE Approved & NAAC Accredited

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that the report entitled, "Programming Virtual Internship Program" is a bonafide record of Internship work done at CISCO Networking Academy during the academic year 2022-2023 by

Pratheesh Raj P L (Reg. No: URK21CS1064)

in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering of Karunya Institute of Technology and Sciences.

ACKNOWLEDGEMENT

First and foremost, I praise and thank ALMIGHTY GOD whose blessings have bestowed in me the willpower and confidence to carry out my Internship.

I am grateful to our beloved founders **Late. Dr. D.G.S. Dhinakaran, C.A.I.I.B, Ph.D** and **Dr. Paul Dhinakaran, M.B.A, Ph.D**, for their love and always remembering us in their prayers.

I extend my thanks to our Vice Chancellor **Dr.P. Mannar Jawahar, Ph.D**and our Registrar **Dr. Elijah Blessing, M.E., Ph.D,** for giving me this opportunity to do the internship.

I would like to thank **Dr. Prince Arulraj, M.E., Ph.D.,** Dean, School of Engineering and Technology for his direction and invaluable support to complete the same.

I would like to place my heart-felt thanks and gratitude to **Dr. J. Immanuel John Raja, M.E., Ph.D.,** Head of the Department, Computer Science and Engineering for his encouragement and guidance.

I feel it a pleasure to be indebted to, **Internship Coordinators**, Department of Computer Science and Engineering and Dr. GetziJeba for their invaluable support, advice and encouragement.

I also thank all the staff members of the Department for extending their helping hands to make this in Internship a successful one.

I would also like to thank all my friends and my parents who have prayed and helped me during the Internship.

INDEX

SNo	Content	PageNo
1.	Introduction 1.1 Company Profile 1.2 Objective of the company 1.3 Organization Chart 1.4 Overview of the Internship 1.5 Chapter-wise Summary	
2.	Nature of the Internship 2.1 Technical Exposure Gained 2.2 Project Details 2.2.1 Objective 2.2.2 Module Details 2.2.3 Design Methodology 2.2.4 Implementation Details 2.2.5 Result Analysis	
3.	Outcomes of the Internship 3.1 Impacts of Internship 3.2 Environment Learning 3.3 Merits and Demerits of Internship 3.4 Applications for Internship	
4.	Conclusions and Future Directions	
5.	References	
6.	Appendixes	

1.1 COMPANY PROFILE

Cisco Systems, Inc. is the worldwide leader in networking for the Internet. Today, networks are an essential part of business, education, government, and home communications. Cisco hardware, software, and service offerings are used to create the Internet solutions that make these networks possible, giving individuals, companies, and countries easy access to information anywhere,at any time. In addition, Cisco has pioneered the use of the Internet in its own business practice and offers consulting services based on its experience to help other organizations around the world.

Cisco was founded in 1984 by a small group of computer scientists from Stanford University. This year, the company celebrates 20 years of commitment to technology innovation, industry leadership, and corporate social responsibility. Since the company's inception, Cisco engineers have led in the innovation of Internet Protocol (IP)-based networking technologies. This tradition of IP innovation continues with the development of industry-leading products in the core technologies of routing and switching, along with Advanced Technologies in areas such as home networking, IP telephony, optical networking, security, storage area networking, and wireless technology. In addition to its products, Cisco provides a broad range of service offerings, including technical support and advanced services.

Cisco sells its products and services, both directly through its own sales force as well as through its channel partners, to large enterprises, commercial businesses, service providers, and consumers. As a company, Cisco operates on core values of customer focus and corporate social responsibility. We express these values through global involvement in educational, community, and philanthropic efforts.

TRANSFORM:-We can transform the fundamental shape of our client business. Regardless of which team you engage with, we have a best-practice process for delivering value. We ensuring that client's receive the business value were we promised.

OPTIMIZE:-Beyond transformation and innovation, it boils down to execution - delivering on time, on budget and "on value". We can optimize your core operations to drive best-in-class efficiency and help fund the transformation and innovation.

INNOVATE:-We can inject a level of product and service innovation into your business to create new revenue opportunities through collaboration and co-creation.

1.2 OBJECTIVE OF THE COMPANY

Cisco enables people to make powerful connections- whether in business, education, philanthropy, or creativity. Cisco hardware, software, and service offerings are used to create the Internet solutions that make networks possible--providing easy access to information anywhere, at any time.

Since the company's inception, Cisco engineers have been leaders in the development of Internet Protocol (IP)-based networking technologies. Today, with more than 71,000 employees worldwide, this tradition of innovation continues with industry-leading products and solutions in the company's core development areas of routing and switching, as well as in advanced technologies such as home networking, IP telephony, optical networking, security, storage area networking, and wireless technology. In addition to its products, Cisco provides a broad range of service offerings, including technical support and advanced services.

Cisco sells its products and services, both directly through its own sales force as well as through its channel partners, to large enterprises, commercial businesses, service providers, and consumers.

The vision statement for Cisco Systems Inc Video is its strategic plan for the future – it defines what and where Cisco Systems Inc Video Company wants to be in the future. The vision statement for Cisco Systems Inc Video is a document identifying the goals of Cisco Systems Inc Video to facilitate its strategic, managerial, as well as general decision making processes.

Mission and Vision of Cisco

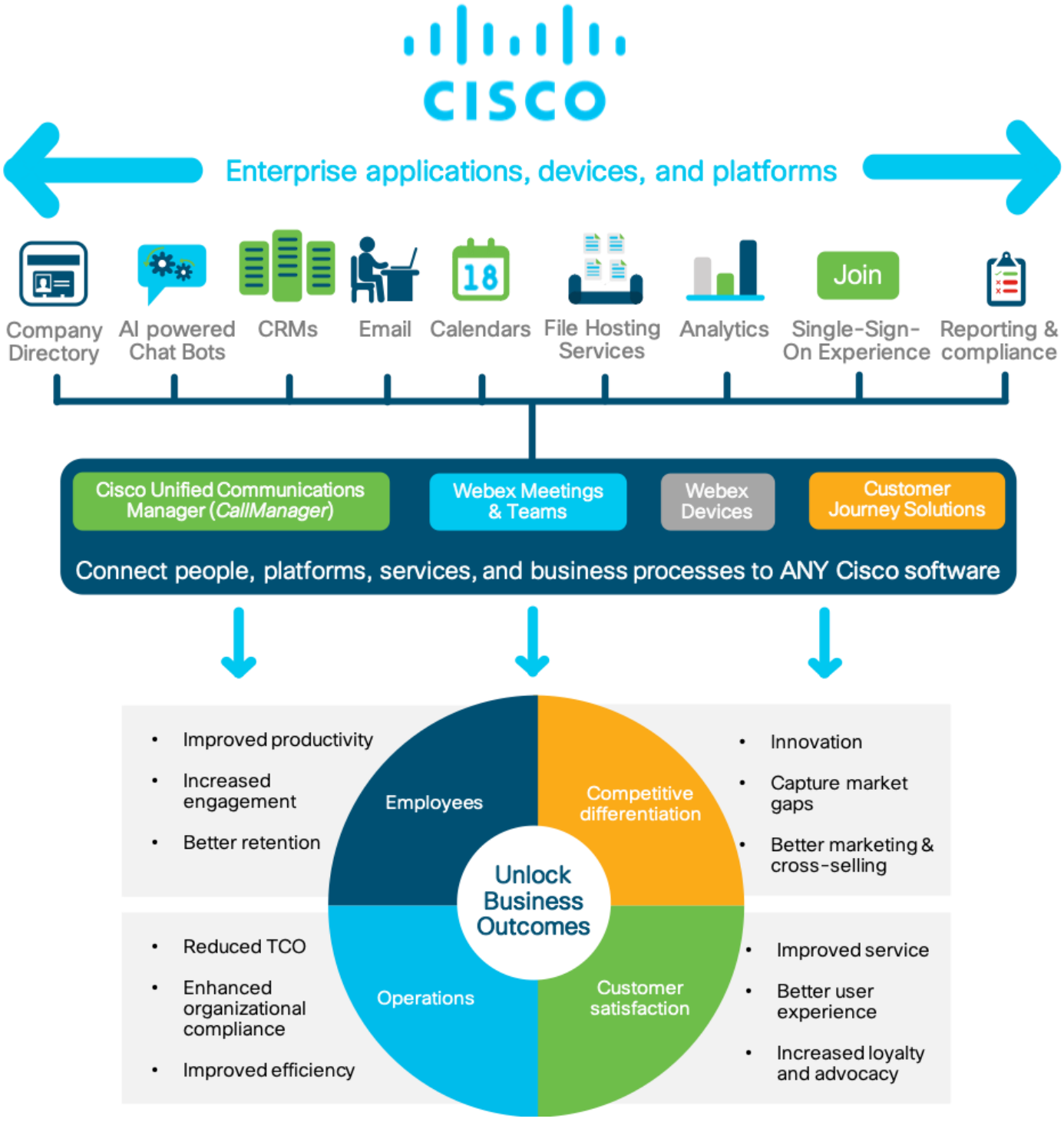
Vision

Cisco's vision is aimed at changing the way we work, live, play, and learn.

Mission

Cisco's mission is to shape the future of the Internet by creating unprecedented value and opportunity for our customers, employees, investors, and ecosystem partners.

1.3 ORGANIZATION CHART



1.4OVERVIEW OF THE INTERNSHIP

Cisco Networking Academy offers the Virtual Internship Program to second- and third-year engineering students to bridge the gap between industry and academia. This is the first time in India where the virtual internship model is being piloted at scale in a collaboration between industry and government for 20,000 students. Virtual internships make it possible to offer experiential learning opportunities to more students.

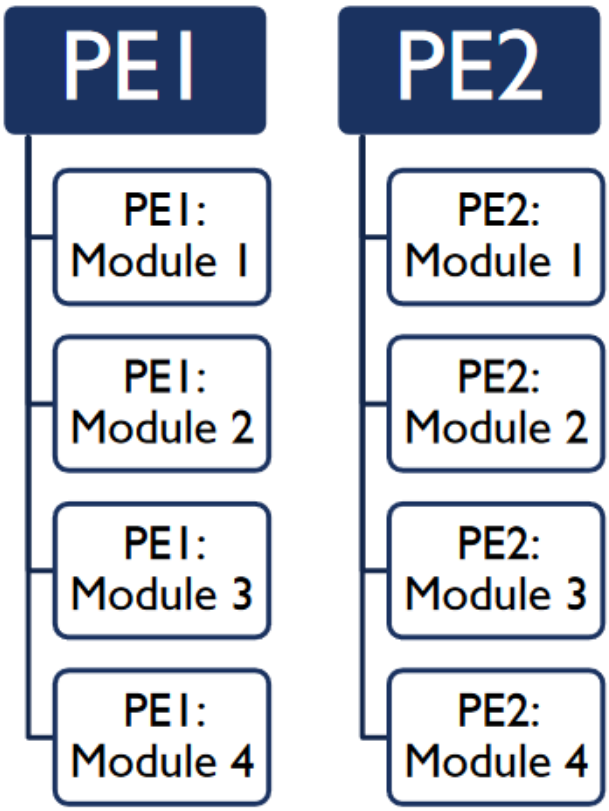
This program redefines experiential learning gained during an internship. In a conventional internship model, students gain knowledge and skills and then work on a problem statement to gain hands-on experience with the guidance of an industry professional.

The VIP is an attempt to virtualize the traditional internship model and allow scalability during this challenging climate. In the current pilot:

- Students gained programming knowledge and skills by completing Cisco Networking Academy courses like Introduction of Cybersecurity and Programming Essentials.
- Next, students learned to use the powerful network simulation tool – Packet Tracer.
- After completing the courses, industry experts and Cisco leaders covered topics from enabling a global skillset to careers in programming.
- Accredited Networking Academy instructors trained faculty members from participating institutions to act as local mentors.
- With the guidance of the local mentor at the institution, students worked on a project to design a interface on interface designs .

1.5 CHAPTER WISE SUMMARY

The PCAP – Programming Essentials in Python course Is divided into two parts. Each part can be taught as an Independent mini-course over a semester.Python Essentials – Part 1 (PE1) is aligned with PCEP certification,while Python Essentials – Part 2 (PE2) is aligned with PCAP certification.Each module concludes with a brief quiz and a Module Test. Additionally, each part (PE1 and PE2) ends with a Summary Test,which includes all the most important questions covered in modules 1 through 4.



CHAPTER 1
Python Language Introduction

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to browsers to games.

HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

PYTHON FEATURES

Python's features include:

• **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

• **Easy-to-read:** Python code is more clearly defined and visible to the eyes.

• **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.

• **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

• **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

• **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

• **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

• **Databases:** Python provides interfaces to all major commercial databases.

• **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

• **Scalable:** Python provides a better structure and support for large programs than shell scripting.

PYTHON ESSENTIALS – PART 1 (PE1)

After completing **PE1 Module 1**, the student will:

- Have a basic knowledge of computer programming and software development;
- Understand the fundamental programming concepts, such as: compiler, interpreter, source code, machine code, IDE;
- Have an orientation in Python's development history, its main traits and features;
- Gain skills allowing her/him to install and configure basic development tools as well as code, and run the very first Python program.

CHAPTER 2

DATA TYPES: It is a set of values, and the allowable operations on those values. It can be one of the following:

Variables can hold values, and every value has a data-type. Python is a dynamically typed language; hence we do not need to define the type of the variable while declaring it. The interpreter implicitly binds the value with its type.

Standard data types

A variable can hold different types of values. For example, a person's name must be stored as a string whereas its id must be stored as an integer.

Python provides various standard data types that define the storage method on each of them. The data types defined in Python are given below.

1. Numbers
2. Sequence Type
3. Boolean
4. Set
5. Dictionary

Number stores numeric values. The integer, float, and complex values belong to a Python Numbers data-type. Python provides the **type()** function to know the data-type of the variable. Similarly, the **isinstance()** function is used to check an object belongs to a particular class.

Python creates Number objects when a number is assigned to a variable. For example;

```
a = 5
print("The type of a", type(a))

b = 40.5
print("The type of b", type(b))

c = 1+3j
print("The type of c", type(c))
print(" c is a complex number", isinstance(1+3j,complex))
```

Python supports three types of numeric data.

1. **Int -** Integer value can be any length such as integers 10, 2, 29, -20, -150 etc. Python has no restriction on the length of an integer. Its value belongs to **int**
2. **Float -** Float is used to store floating-point numbers like 1.9, 9.902, 15.2, etc. It is accurate upto 15 decimal points.
3. **complex -** A complex number contains an ordered pair, i.e., x + iy where x and y denote the real and imaginary parts, respectively. The complex numbers like 2.14j, 2.0 + 2.3j, etc.

SEQUENCE TYPE

String

The string can be defined as the sequence of characters represented in the quotation marks. In Python, we can use single, double, or triple quotes to define a string.

String handling in Python is a straightforward task since Python provides built-in functions and operators to perform operations in the string.

In the case of string handling, the operator + is used to concatenate two strings as operation

The operator * is known as a repetition operator as the operation "Python" *2 returns 'Python Python'.

The following example illustrates the string in Python.

Example - 1

```
str = "string using double quotes"
print(str)
s = """A multiline
string"'
print(s)
```

List

Python Lists are similar to arrays in C. However, the list can contain data of different types. The items stored in the list are separated with a comma (,) and enclosed within square brackets [].

We can use slice [:] operators to access the data of the list. The concatenation operator (+) and repetition operator (*) works with the list in the same way as they were working with the strings.

```
list1  = [1, "hi", "Python", 2]
#Checking type of given list
print(type(list1))

#Printing the list1
print (list1)

# List slicing
print (list1[3:])
```

```
# List slicing
print (list1[0:2])

# List Concatenation using + operator
print (list1 + list1)
```

Dictionary

Dictionary is an unordered set of a key-value pair of items. It is like an associative array or a hash table where each key stores a specific value. Key can hold any primitive data type, whereas value is an arbitrary Python object.

The items in the dictionary are separated with the comma (,) and enclosed in the curly braces {}.

Consider the following example.

```
d = {1:'Jimmy', 2:'Alex', 3:'john', 4:'mike'}

# Printing dictionary

print (d)

# Accesing value using keys

print("1st name is "+d[1])

print("2nd name is "+ d[4])


print (d.keys())

print (d.values())
```

Boolean

Boolean type provides two built-in values, True and False. These values are used to determine the given statement true or false. It denotes by the class bool. True can be represented by any non-zero value or 'T' whereas false can be represented by the 0 or 'F'. Consider the following example.

```
# Python program to check the boolean type

print(type(True))

print(type(False))
```

SET

Python Set is the unordered collection of the data type. It is iterable, mutable(can modify after creation), and has unique elements. In set, the order of the elements is undefined; it may return the changed sequence of the element. The set is created by using a built-in function **set(),** or a sequence of elements is passed in the curly braces and separated by the comma. It can contain various types of values. Consider the following example.

```
# Creating Empty set
set1 = set()
set2 = {'James', 2, 3,'Python'}
#Printing Set value
print(set2)
# Adding element to the set
set2.add(10)
print(set2)
#Removing element from the set
set2.remove(2)
print(set2)
```

After completing **PE1 Module 2**, the student will:

- Gain skills enabling her/him to create, edit and run Python source files using IDLE;
- Have some knowledge of Python's numeral literals, their syntax, types and formats;
- Have an orientation in issues related to Python arithmetic operators and expressions;
- Gain the ability to name, create, initialize and modify variables;

Have skills that enable her/him to perform basic input/output operations in a Python program.

CHAPTER 3

PYTHON OPERATORS

Operators are used to perform operations on variables and values. In the example below, we use the + operator to add together two values: **Example**

print(10 + 5)

Python divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Arithmetic Operators

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 30
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -10
*Multiplication	Multiplies values on either side of the operator	a * b = 200
/ Division	Divides left hand operand by right hand operand	b / a = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 0
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity)	9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0

Assignment Operators

Assignment operators are used to assign values to variables:

Operator	Description	Example
=	Assigns values from right side operands to left side operand	x = 5
+= Add AND	It adds right operand to the left operand and assign the result to left operand	x += 3 (same as) x = x + 3
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	x -= 3 (same as) x = x - 3
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	x *= 3 (same as) x = x * 3
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	x /= 3 (same as) x = x / 3

Python Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Python Membership Operators

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

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Python Bitwise Operators

Bitwise operators are used to compare (binary) numbers:

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
|	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>		Shift right by pushing copies of the leftmost bit in on the left, and let the rightmost bits fall off

After completing **PE1 Module 3**, the student will:

- Know basic features of the Boolean data type;
- Gain skills to work with relational operators in Python;
- Have the ability to effectively use the control statements if, if-else and if-elif-else;
- Understand the role of a loop and be able to use the control statements while and for;
- Have an orientation in bitwise operations in Python;
- Know the role of lists and be able to operate with them to perform actions including indexing, slicing and content manipulation;
- Understand how the bubble-sort algorithm works;
- Have a knowledge of multidimensional lists in Python.

CHAPTER 4
FUNCTIONS IN PYTHON

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing. Python gives you many built-in functions like print(), etc. but you can also create your own functions. These functions are called user-defined functions.

Defining a Function

Simple rules to define a function in Python.

Function blocks begin with the keyword def followed by the function name and parentheses (( )).

Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.

The first statement of a function can be an optional statement - the documentation string of the function or docstring.

The code block within every function starts with a colon (:) and is indented.

The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

```
def functionname( parameters ):
"function_docstring"
function_suite
return [expression]
```

Calling a Function

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt. Following is the example to call printme() function −

```
# Function definition is here
def printme( str ):
"This prints a passed string into this function"
print str
return;
```

```
# Now you can call printme function
printme("I'm first call to user defined function!")
printme("Again second call to the same function")
```

Function Arguments

You can call a function by using the following types of formal arguments:

• Required arguments

• Keyword arguments

• Default arguments

• Variable-length arguments

Scope of Variables

All variables in a program may not be accessible at all locations in that program. This depends on where

you have declared a variable.

The scope of a variable determines the portion of the program where you can access a particular identifier.

There are two basic scopes of variables in Python −

1 Global variables

2 Local variables

Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope.

This means that local variables can be accessed only inside the function in which they are declared, whereas global variables can be accessed throughout the program body by all functions. When you call a function, the variables declared inside it are brought into scope.

After completing **PE1 Module 4**, the student will:

- Understand the concept of functions and be able to code and invoke her/his own functions;
- Have an orientation of the main features of structural programming;
- Have some knowledge of name scopes and be able to distinguish global and local variables, as well as understand how name shadowing works;

- Understand the principles of tuples including the immutability notion;
- Know the role of dictionaries and be able to use them effectively in appropriate circumstances;
- Be ready to take the Part 1 Summary Test, and attempt the qualification PCEP – Certified Entry-Level Python Programmer from the OpenEDG Python Institute.

PYTHON ESSENTIALS – PART 2 (PE2)

Modules, Packages, and PIP

INTRODUCTION

Nowadays, the Python programming language becomes one of the most popular languages. When we write the codes for Production level Data Science Projects, what happens is that our Python code grows in size, and as a result most probably it becomes unorganized over time. So, keeping your code in the same file as it grows makes your code difficult to maintain and debug.

So, to resolve these kinds of issues, Python modules help us to organize and group the content by using files and folders. This modular programming approach where we have broken the code into separate parts is where python modules come into the picture. So, In this article, I will help you to understand the complete intuition behind modules in Python in a detailed manner.

The topics which we are going to discuss in this article are as follows:

- What are Python Modules
- How to create Python Modules
- How to use Python Modules
- How we can introduce variables in Python Modules
- How to rename a Python Module
- How does import from modules work
- What are the advantages to use modules in Python
- Working with Built-in Modules in Python – Math, and Statistics Module.

PYTHON MODULES

In Python, **Modules** are simply files with the ".py" extension containing Python code that can be imported inside another Python Program.

In simple terms, we can consider a module to be the same as a code library or a file that contains a set of functions that you want to include in your application.

With the help of modules, we can organize related functions, classes, or any code block in the same file. So, It is considered a best practice while writing bigger codes for production-level projects in Data Science is to split the large Python code blocks into modules containing up to 300–400 lines of code.

The module contains the following components:

- Definitions and implementation of classes,
- Variables, and
- Functions that can be used inside another program.

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference. Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.

Example:

The Python code for a module named *aname*normally resides in a file named *aname.py*.

Here's an example of a simple module, support.py

```
def print_func( par ):
print "Hello : ", par
return
```

How to create Python Modules

To create a module, we have to save the code that we wish in a file with the file extension **".py"**. Then, the name of the Python file becomes the name of the module.

For Example,

In this program, a function is created with the name **“welcome”** and save this file with the name **mymodule.py**i.e. name of the file, and with the extension **“.py”**.

We saved the following code in a file named **mymodule.py**

```
def welcome(name):
print("Hello, " + name +" to Analytics Vidhya")
```

How to use Python Modules

To incorporate the module into our program, we will use the **import keyword,** and to get only a few or specific methods or functions from a module, we use the **from keyword.**

Variables in Python Modules

The module can contain functions, as already described, but can also contain variables of all types such as arrays, dictionaries, objects, etc.

For Example,

Save this code in the file **mymodule.py**

```
person1 = {
  "name": "Chirag Goyal",
  "age": 19,
  "country": "India"
  "education”: “IIT Jodhpur"
}
```

Packages in Python

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-sub packages.

Consider a file Pots.py available in Phone directory. This file has following line of source code − def Pots():

Similar way, we have another two files having different functions with the same name as above −

• Phone/Isdn.py file having function Isdn()

• Phone/G3.py file having function G3()

Now, create one more file __init__.py in Phone directory −

• Phone/__init__.py

To make all of your functions available when you've imported Phone,to put explicit import statements in

__init__.py as follows −

```
from Pots import Pots
from Isdn import Isdn
from G3 import G3
```

After you add these lines to __init__.py, you have all of these classes available when you import the

Phone package.

```
# Now import your Phone Package.
import Phone
Phone.Pots()
Phone.Isdn()
Phone.G3()
```

Difference between Module and Package in Python

The main difference between a module and a package in Python is that a module is a simple Python script with a .py extension file that contains collections of functions and global variables while a package is a directory that contains a collection of modules, and this directory also contains an __init__.py file by which the interpreter interprets it as a package.

To create large-scale-based real-world applications, we divide large code into smaller pieces to perform different functionalities, which ultimately results in a large number of modules. To collaborate with all of the modules, we create a Python package with an __init__.py file that informs the Python Interpreter that the given folder is a Python Package.

For any source code, a Python package serves as a user-variable interface. This functionality enables any functional runtime script to use a Python package at a specified moment, this shows the main difference between module and package in Python.

What Makes Python Package Different from Modules

When using a library, a Python package defines the code as a separate unit for each function. While the modules themselves are a distinct library with built-in functionality, the advantage of packages over modules is their **reusability**. So this is the difference between module and package in Python.

Let's summarise our discussion on the difference between module and package in Python by mentioning some of the important points.

- A **Python Module** can be a simple python File (**.py** extension file) i.e a combination of numerous Functions and Global variables.
- A **Python Package** is a collection of different Python modules with an __init__.py File.
- __init__.py Python File works as a Constructor for the Python Package.
- Python Packages and Modules support functionalities like Explicit Namespace and Convenience API.
- If we talk about the basic difference between module and package in Python. A Python package serves as a user-variable interface, whereas Python modules serve as a ready-made library.

After completing **PE2 Module 1**, the student will:

- Understand the role of the Python module and know the available ways of importing modules into her/his own code/name space.
- Gain knowledge of selected useful standard Python modules;
- Have an orientation in package purposes as well as be able to create her/his own packages.
- Know the main function of PIP and be able to use it in order to install and uninstall ready-to-use packages from PyPI.

In Python, all exceptions must be instances of a class that derives from Base Exception. In a try statement with an except clause that mentions a particular class, that clause also handles any exception classes derived from that class (but not exception classes from which *it* is derived). Two exception classes that are not related via subclassing are never equivalent, even if they have the same name.

The built-in exceptions listed below can be generated by the interpreter or built-in functions. Except where mentioned, they have an “associated value” indicating the detailed cause of the error. This may be a string or a tuple of several items of information (e.g., an error code and a string explaining the code). The associated value is usually passed as arguments to the exception class’s constructor.

Python String Methods

A string is a sequence of characters enclosed in quotation marks. In this reference page, you will find all the methods that a string object can call. For example, you can use the join() method to concatenate two strings.

capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string
index()	Searches the string for a specified value and returns the position of where it was found

Python List Methods

Python has a lot of list methods that allow us to work with lists. In this reference page, you will find all the list methods to work with Python lists. For example, if you want to add a single item to the end of the list, you can use the list.append() method.

Python List append()	Adds an element at the end of the list
Python List clear()	Removes all the elements from the list
Python List copy()	Returns a copy of the list
Python List count()	Returns the number of elements with the specified value
Python List extend()	Add the elements of a list (or any iterable), to the end of the current list
Python List index()	Returns the index of the first element with the specified value
Python List insert()	Adds an element at the specified position
Python List pop()	Removes the element at the specified position
Python List remove()	Removes the first item with the specified value
Python List reverse()	Reverses the order of the list
Python List sort()	Sorts the list

Strings vs. lists – Similarities and Differences

Strings and lists share many similarities as we have seen throughout this lesson. However, strings are not interchangeable with lists because of some important differences.

Strings can only consist of characters, while lists can contain any data type.

Because of the previous difference, we cannot easily make a list into a string, but we can make a string into a list of characters, simply by using the `list()` function.

```
message = "Hello"
message_to_list = list(message)
print(message_to_list)
# message_to_list is now ['H', 'e', 'l', 'l', 'o']
```

Furthermore, when we add to a string, we use the + concatenation operator to do so. With a list, we use the `.append()` method to add elements.

Finally, one of the most important differences is mutability. Strings are immutable, meaning that we cannot update them. We could update values in a list however quite easily.

After completing **PE2 Module 2**, the student will:

- Know how characters are coded and stored inside the computer's memory, distinguish most known coding standards;
- Gain knowledge of Python's sequences and know the differences between strings and lists;
- Be able to effectively use selected lists and string methods;
- Have an orientation of Python's way of identifying and handling runtime errors;
- Understand the purpose of the control statements try, except and raise;
- Understand Python exception hierarchies.

CHAPTER 5
Object-Oriented Programming

Python has been an object-oriented language since it existed. Because of this, creating and using classes and objects are downright easy. Python is a great programming language that supports OOP. We use it to define a class with attributes and methods. This chapter helps to become an expert in using Python's object oriented programming support.

Overview of OOP Terminology

• **Class:** A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.

• **Class variable:** A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.

• **Data member:** A class variable or instance variable that holds data associated with a class and its objects.

• **Function overloading:** The assignment of more than one behavior to a particular function. The operation performed varies by the types of objects or argument

• **Instance variable:** A variable that is defined inside a method and belongs only to the current instance of a class.

• **Instance:** An individual object of a certain class. An object obj that belongs to a class Circle, for example, is an instance of the class Circle.

• **Instantiation:** The creation of an instance of a class.

• **Object:** A unique instance of a data structure that's defined by its class. An object comprises both data members (class variables and instance variables) and methods.

• **Operator overloading:** The assignment of more than one function to a particular operator.

Method

The method is a function that is associated with an object. In Python, a method is not unique to class instances. Any object type can have methods.

Inheritance

Inheritance is the most important aspect of object-oriented programming, which simulates the real-world concept of inheritance. It specifies that the child object acquires all the properties and behaviors of the parent object.

By using inheritance, we can create a class which uses all the properties and behavior of another class. The new class is known as a derived class or child class, and the one whose properties are acquired is known as a base class or parent class.

It provides the re-usability of the code.

Polymorphism

Polymorphism contains two words "poly" and "morphs". Poly means many, and morph means shape. By polymorphism, we understand that one task can be performed in different ways. For example - you have a class animal, and all animals speak. But they speak differently. Here, the "speak" behavior is polymorphic in a sense and depends on the animal. So, the abstract "animal" concept does not actually "speak", but specific animals (like dogs and cats) have a concrete implementation of the action "speak".

Encapsulation

Encapsulation is also an essential aspect of object-oriented programming. It is used to restrict access to methods and variables. In encapsulation, code and data are wrapped together within a single unit from being modified by accident.

Data Abstraction

Data abstraction and encapsulation both are often used as synonyms. Both are nearly synonyms because data abstraction is achieved through encapsulation.

Abstraction is used to hide internal details and show only functionalities. Abstracting something means to give names to things so that the name captures the core of what a function or a whole program does.

After completing **PE2 Module 3**, the student will:

- Understand the fundamental concepts of object programming like class, object, property, method, inheritance and polymorphism;
- Have an orientation in the differences between procedural and object approaches, as well as being oriented when both of the techniques reveal their pros and cons;
- Be able to build her/his own classes, objects, properties and methods;
- Be able to use inheritance and polymorphism in her/his inheritance path;
- Understand the objective nature of Python exceptions.

Chapter 6 Miscellaneous

The Python Standard Library is a collection of exact syntax, token, and semantics of Python. It comes bundled with core Python distribution. We mentioned this when we began with an introduction.

It is written in C, and handles functionality like I/O and other core modules. All this functionality together makes Python the language it is.

More than 200 core modules sit at the heart of the standard library. This library ships with Python.

After completing **PE2 Module 4**, the student will:

- Gain the ability to understand the concepts of generators, iterators and closures as well as be able to use them in adequate applications;
- Know how Python accesses physical file-system resources, understand file open modes and perform basic input/output Operations in relation to text and binary files;
- Gain an ability to manipulate date and time, work with a calendar, and create directory structures using Python;
- Be ready to take the Part 2 Summary Test, and the Final Test;
- Be prepared to attempt the qualification PCAP – Certified Associate in Python Programming from the OpenEDG Python Institute.

NATURE OF INTERNSHIP

TECHNICAL EXPOSURE

The four weeks of our internship provided a great experience in the field of installation, programming, logical and servicing of Cisco systems. Python is a user friendly programming language and it plays a significant role in the I.T industry as it is flexible, open-source and it is required by companies in higher demand.

This internship was very useful to gain experience in the real technical world and also know the job responsibility. This internship was very useful to gain experience in the real technical world and also know the job responsibility. This internship program was helpful to learn various professional skills like logical skill, technical skill and presentation skills etc, which are most essential to serve organization in future

OBJECTIVE

- Internships are generally thought of to be reserved for college students looking to gainexperience in a particular field. However, a wide array of people can benefit fromTraining Internships in order to receive real world experience and develop their skills.
- An objective for this position should emphasize the skills you already possess in the areaand your interest in learning more
- Internships are utilized in a number of different career fields, including architecture, engineering, healthcare, economics, advertising and many more.
- Some internship is used to allow individuals to perform scientific research while othersare specifically designed to allow people to gain first-hand experience working.
- Utilizing internships is a great way to build your resume and develop skills that can beemphasized in your resume for future jobs. When you are applying for a TrainingInternship, make sure to highlight any special skills or talents that can make you standapart from the rest of the applicants so that you have an improved chance of landing theposition.

PROJECT DETAILS

To convert the Ipv4 address to binary,hex and octal values and put it on the next file and the data will be present in the text file and the input should be in the run time.

PROBLEM STATEMENT:

Task 1 : Ask the user to input 10 ipv4 addresses.

Task 2 : Check if the addresses are valid ipv4 addresses.

Task 3 : Convert the ipv4 addresses which are in decimal format to Binary, Octal and Hexadecimal format. For conversion use function (inbuilt or library).

Task 4 : Create a list which will hold the addresses . [Decimal, Binary, Octal and Hexadecimal].

Task 5 : Transfer the contents of the list to file named conversion.text.

Task 6: Print the following output on the screen (O/P OF TASK 4).

Description:

Decimal to binary conversion is an important task to understand in IP addressing and Subnetting. IP addressing is a core functionality of networking today. The knowledge of how to assign an IP address, or determine the network or host ID via a subnet, is vital to any good network engineer. Having a good, solid understanding of the simple things makes more complex tasks easier. Here are steps on how to convert a decimal IP address to its binary form, without memorization.

Source Code:

```
# Containers
IPList = []  # List of IP Addresses in Decimal
BinaryIPList = [] # List of IP Addresses in Binary
OctalIPList = [] # List of IP Addresses in Octal
HexadecimalIPList = [] # List of IP Addresses in Hexadecimal

# Step 1: Take Valid IP addresses in IPList

print("How many IPv4 Addresses you want to enter?")
N = int(input())
print("Enter", N, "IPv4 Addresses")
i = 0
while( i< N ):
   IP = input()
OctateCount = 0
octate = ""
```

```
DotCount = 0
    for j in range(0, len(IP)+1):
if( j == len(IP) or IP[j] == '.'):
octate = int(octate)

if(j != len(IP) and IP[j] == '.'):
DotCount+=1

if(octate> 255 or octate< 0 or DotCount> 3):
print(" Not a Valid IPv4 Address Try Again")
            break
else:
OctateCount+=1
octate = ""
if(OctateCount == 4):
IPList.append(IP)
print("Valid Enter Another")
i += 1
        else:
octate = str(octate) + str(IP[j])
```

```
# Step 2 Convert IP Addresses from Decimal to Binary and Populate BinaryIPList

for i in IPList:
octate = ""
BinaryIP = ""
    for j in range(0, len(i)+1):
        if (j != len(i) and i[j] == '.'):
octate = int(octate)
BinaryIP += str(bin(octate)[2:]) + '.'
octate =""
elif (j == len(i) ):
octate = int(octate)
BinaryIP += str(bin(octate)[2:])
octate = ""
BinaryIPList.append(BinaryIP)
        else:
octate = str(octate) + str(i[j])
```

```
# Step 3 Convert IP Addresses from Decimal to Octal and Populate OctalIPList
```

```
for i in IPList:
octate = ""
OctalIP = ""
   for j in range(0, len(i)+1):
      if (j != len(i) and i[j] == '.'):
octate = int(octate)
OctalIP += str(oct(octate)) + '.'
octate =""
elif (j == len(i) ):
octate = int(octate)
OctalIP += str(oct(octate))
octate = ""
OctalIPList.append(OctalIP)
      else:
octate = str(octate) + str(i[j])
```

```
# Step 4 Convert IP Addresses from Decimal to Hexadecimal & Populate Hexadecimal IPList

for i in IPList:
octate = ""
HexadecimalIP = ""
   for j in range(0, len(i)+1):
      if (j != len(i) and i[j] == '.'):
octate = int(octate)
HexadecimalIP += str(hex(octate)) + '.'
octate =""
elif (j == len(i) ):
octate = int(octate)
HexadecimalIP += str(hex(octate))
octate = ""
HexadecimalIPList.append(HexadecimalIP)
      else:
octate = str(octate) + str(i[j])
```

```
# Step 5 Transfer IP addresses in a file named conversion.txt

file = open("conversion.txt", 'w')

file.write("Decimal IP Addresses are as follows:\n")
for i in IPList:
file.write(i)
file.write('\n')

file.write('\n')
file.write("Binary conversion of IP Addresses are as follows:\n")
for i in BinaryIPList:
file.write(i)
file.write('\n')

file.write('\n')
file.write("Hexadecimal conversion of IP Addresses are as follows:\n")
for i in HexadecimalIPList:
file.write(i)
file.write('\n')

file.close()

#Final Step Print contents of all the IP Lists.

print(IPList)

print(BinaryIPList)
```

file.write('\n') file.write('\n') file.write("Octal conversion of IP Addresses are as follows:\n") for i in OctalIPList: file.write(i)	print(OctalIPList) print(HexadecimalIPList)

IMPLEMENTATION DETAILS

- The PCAP: Programming Essentials in Python curriculum is designed for students with little or no prior knowledge of programming.
- The PCAP: Programming Essentials in Python course covers all the basics of programming in Python, as well as general computer programming concepts and techniques.
- The course also familiarizes the student with the object-oriented approach.
- The PCAP: Programming Essentials in Python (2.0) curriculum helps students prepare for the PCEP: Certified Associate in Python Programming (PE1: Modules 1-4) and PCAP: Python Certified Associate Programmer (PE2: Modules 1-4) certification exams.
- PCEP – Certified Entry-Level Python Programmer certification is an optional interim step to the PCAP – Certified Associate in Python Programming certification.

Result Analysis

PCAP – Certified Associate in Python Programming certification is a professional credential that measures the ability to accomplish coding tasks related to the basics of programming in the Python language, and the fundamental notions and techniques used in object-oriented programming.

Output:

```
How many IPv4 Addresses you want to enter?
10
Enter 10 IPv4 Addresses
1.1.1.1
Valid Enter Another
23.56.78.90
Valid Enter Another
233.199.65.1
Valid Enter Another
15.1.3.4
Valid Enter Another
34.65.21.89
Valid Enter Another
90.12.43.65
Valid Enter Another
89.55.2.1
Valid Enter Another
78.65.4.3
Valid Enter Another
89.54.21.3
Valid Enter Another
78.90.40.21
Valid Enter Another

Valid ip:
1.1.1.1
23.56.78.90
233.199.65.1
15.1.3.4
34.65.21.89
90.12.43.65
89.55.2.1
78.65.4.3
89.54.21.3
78.90.40.21

BinaryIPList:
1.1.1.1
10111.111000.1001110.1011010
11101001.11000111.1000001.1
1111.1.11.100
100010.1000001.10101.1011001
1011010.1100.101011.1000001
1011001.110111.10.1
1001110.1000001.100.11
1011001.110110.10101.11
1001110.1011010.101000.10101

OctalIPList:
0o1.0o1.0o1.0o1
0o27.0o70.0o116.0o132
0o351.0o307.0o101.0o1
0o17.0o1.0o3.0o4
0o42.0o101.0o25.0o131
0o132.0o14.0o53.0o101
0o131.0o67.0o2.0o1
0o116.0o101.0o4.0o3
0o131.0o66.0o25.0o3
0o116.0o132.0o50.0o25

HexadecimalIPList:
0x1.0x1.0x1.0x1
0x17.0x38.0x4e.0x5a
0xe9.0xc7.0x41.0x1
0xf.0x1.0x3.0x4
0x22.0x41.0x15.0x59
0x5a.0xc.0x2b.0x41
0x59.0x37.0x2.0x1
0x4e.0x41.0x4.0x3
0x59.0x36.0x15.0x3
0x4e.0x5a.0x28.0x15

Process finished with exit code 0
```

OUTCOMES OF INTERSHIP

IMPACTS OF INTERSHIP

- It is omnipresent – people use numerous Python-powered devices on a daily basis, whether they realize it or not.
- There have been millions (well, actually billions) of lines of code written in Python, which means almost unlimited
- opportunities for code reuse and learning from well-crafted examples.
- It is easy to learn – the time needed to learn Python is shorter than for many other languages; this means that it's possible
- to start the actual programming faster.
- It is easy to use for writing new software – it's often possible to write code faster when using Python.
- It is easy to obtain, install and deploy – Python is free, open and multiplatform; not all languages can boast that.
- There is a large and very active Python community.
- It gives you a solid foundation and allows you to learn other programming languages (e.g., C++, Java, or C) much easier and much fast.

ENVIRONMENT LEARNING

Key behavior for interns to demonstrate in any work environment are effort, ability to learn and adapt, enthusiasm, self-motivation, and an appropriate level of curiosity. As an intern, enthusiastically approach every task you are given. Even the tasks which may feel beneath your abilities. Learning what drives your career passions is as valuable as learning what brings you down.

Python is on demand availability of computer system resources, especially data storage and computing power, without direct active management by the user. Using python users are able to access software and application from wherever they need.It is a very powerful and complex tool, but it is built on a few core concepts. These concepts are recurring across Sales force products, documentation and customer service and inform the user experience.

- I believe the course/internship has shown conclusively that it is both possible and desirable to use Python as the principal teaching language.
- It is Free (as in both cost and source code).
- It is a flexible tool that allows both the teaching of traditional procedural programming and modern OOP; It can be used to teach a large number of transferable skills;
- It is a real-world programming language that can be and is used in academia and the commercial world.
- It appears to be quicker to learn and, in combination with its many libraries, this offers the possibility of more rapid student development allowing the course to be made more challenging and varied.
- It is trivial to install on a Windows PC allowing students to take their interest further. For many the hurdle of installing a Pascal or C compiler on a Windows machine is either too expensive or too complicated and most importantly, its clean syntax offers increased understanding and enjoyment for students.

Merits And Demerits Of Internship

Merits

- Academic institutions can use this course as follows:
- Offer the course as a complete two-semester course (or offer PE1/PE2 only as a one-semester course)
- Create interest and motivate new students to learn the fundamentals of computer programming;
- Motivate those students who already know another programming language to learn Python.
- Supplement an existing Python language course;
- Help students prepare for the PCEP – Certified Entry-Level Python Programmer and PCAP – Certified Associate in Python Programming certifications;
- Introduce NetAcad and Edube Interactive to peers and colleagues.
- Flexibility: Another significant benefit of Cisco virtual internship is the flexibility of work hours

- Cisco Virtual internship promote student independence. With their help one needs to learn how to work without supervision. We can work freely at our convenience, meet all your deadlines and embrace personal productivity.
- Virtual internships occur remotely. This means any student from any location in the world can attend.

Demerits

- Cisco Online internships don't have human interaction since we work alone in a remote location. Interns communicate with their supervisors through emails, zoom meetings, or phone calls, but never in person.
- Thus, we may feel detached from the organization. That's because you can't meet colleagues and attend company events. This means we can't get first-hand professional experience.
- Virtual internships limit students to create strong networks that can assist them in their career path. An office-based internship allows you to meet with many people in different professions. But with remote working, lack of community networking is what might prevent us from reaching our greatest potential.

APPLICATIONS OF INTERNSHIP

- To become a creator: a highly creative and powerful one. Go as far as your imagination lets you.
- Strong programming skills are a hot commodity on the job market!
- Boost your earning potential.
- Programming is the language of the future.
- Learning to program means learning to think in abstract and more precise ways.

DETAILS OF TRAINING UNDERGONE

- This is a project-based course for beginners to become professional Python programmers. Comprising 30 lectures spanning just above 30 hours in total length.
- This course is ideal for people who have zero experience in the programming world.
- The course is replete with quizzes and exercises that help with understanding the theoretical and practical aspects of Python programming.

The program starts with a complete introduction to Python basics and gradually builds up from there.

CONCLUSION

Practical knowledge means the visualization of the knowledge, which we read in our books. For this, we perform experiments and get observations. Practical knowledge is very important in every field. One must be familiar with the problems related to that field so that he may solve them and become a successful person.

Python is on demand availability of computer system resources, especially data storage and computing power, without direct active management by the user. Using python users are able to access software and application from wherever they need. It is a very powerful and complex tool, but it is built on a few core concepts. These concepts are recurring across Sales force products, documentation and customer service and inform the user experience.

After achieving the proper goal in life, an engineer has to enter in professional life. According to this life, we have to serve an industry, may be public or private sector or self-own. For the efficient work in the field, he must be well aware of the practical knowledge as well as theoretical knowledge.

Due to all above reasons and to bridge the gap between theory and practical, our Engineering curriculum provides a practical training of 30 days. During this period a student work in the industry and get well all type of experience and knowledge about the working of companies and hardware and software tools. I have undergone my 30 days summer training in 2nd sem at Netacad . This report is based on the knowledge, which I acquired during my 30 days of summer training.

REFERENCES

[1] A. Berson and S. Smirth ,*Transients in Power Systems*, 3rd Edition, McGraw Hill, 1998, ISBN:81-265-0280-0.

[2] Larry M.Stephens, "Consensus Ontologies in Web Page Design", *IEEE Transactions On Pattern Recognition and Machine Intelligence*, vol.29, no.4, pp. 120-135, September 1999.

[3] K.E.Barner, "Joint Region Merging Criteria", *Proceedings of International Conference on Image Processing*,vol.25, pp. 108-111, Sept.2007.

[4] [online] http://www.vicomsoft.com/knowledge/reference/firewalls1.html.