#### A Internship Report on

PYTHON

***Submitted to***

*DEPARTMENT OF ELECRONICS AND COMMUNICATION ENGINEERING NARAYANA ENGINEERING COLLEGE, GUDUR*

In partial fulfillment of requirement for the award of the degree of

Bachelor of Technology

In

Electronics & Communication Engineering

*By*

# K.CHAITHANYA [PRABHA

# 20F11A04B9

Under the guidance of

Dr.B.Malakonda Reddy,M.Tech,PhD

professor-Department of ECE



## DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NARAYANA ENGINEERING COLLEGE :: GUDUR

**(An Autonomous College Under JNTUA)**

**2024**

NARAYANA ENGINEERING COLLEGE :: GUDUR

**(An Autonomous College Under JNTUA )**

# 2024

## DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING



**BONAFIDE CERTIFICATE**

This is to certify that the internship entitled “**Python”** is being submitted by K.Chaithanya prabha (20F11A04B9) in partial fulfillment of the requirements for the award of degree of **Bachelor of Technology** in **ELECTRONICS AND COMMUNICATION ENGINEERING, NARAYANA ENGINEERING COLLEGE, GUDUR, an Autonomous college Under** JNTUA Ananthapuramu is recorded to be the bonafide work carried out by her under my guidance and supervision.

**Guide Head of the Department**

Dr.B.Malakonda Reddy,M.Tech,Ph.D Dr.P.Sreenivasulu, M.Tech., Ph.D

Professor Professor

Department of ECE Department of ECE

Narayana Engineering College Narayana Engineering College

# ACKNOWLEDGEMENT

We extremely thankful to **Dr. P. Narayana**, the **Founder Chairman** of **Narayana Group** for his good initiation in starting technical institution in Gudur like rural area for helping economically backward students.

We also thankful to **Mr. K. Puneeth**, the **Chairman** of **Narayana Group** for providing the infrastructural facilities, without these facilities the work would not have been possible.

We would like to express our deep sense of gratitude to **Dr. V. Ravi Prasad, Principal, Narayana Engineering College, Gudur** for his continuous effort in creating a competitive environment in our college and encouraging throughout this program.

We would like to express our deep sense of gratitude to **Dr. K. Vishwaksena Reddy, Dean of Narayana Engineering College, Gudur** for his kind encouragement in doing this project and for completing it successfully.

We would like to convey our heartfelt thanks to **Dr. P. Sreenivasulu, Professor & HOD** of Electronics and Communication Engineering and our sincere thanks to internship Coordinator **Mrs.P.Manjulamma** for giving the opportunity to embark up on this topic and for his continues encouragement throughout the internship.

We would like to thank our Guide **Dr.B.Malakonda Reddy,M.Tech,Ph.D, Professor** Department of ECE**,** for their valuable guidance, constant assistance, support, endurance and constructive suggestions for the betterment of the internship.

We also wish to thank all the **Faculty** of the Department of Electronics & Communication Engineering for helping us directly or indirectly in completing this internship successfully.

We would like to thank the **Non-Teaching Staff** and **Library Staff** for the co-operation in completion of our inteenship.

Finally we are thankful to our **Parents** and **Friends** for their continued moral and material support throughout the course.

**DECLARATION**

I am here by declare that the internship entitled **Python** has been done by me under the guidance of **Dr.B.Malakonda Reddy,M.Tech,Ph.D,**  **Professor, Department of Electronics and Communication Engineering**. This Internship work has been submitted to **NARAYANA ENGINEERING COLLEGE, GUDUR** as a part of partial fulfillment of the requirements for the award of degree of **Bachelor of Technology**.

I am declare that this internship report is not copied in part or whole or otherwise plagiarized the work of other students and/ or persons/ or any entity.

I am declare that this internship report has not been submitted at any time to another institute or University for the award of any degree.

#### K.Chaithanya prabha

#### (20F11A04B9)

**Place: Gudur**

**Date:**

# ABSTRACT

This internship program offers a comprehensive dive into the world of Python programming, focusing on practical applications and experiential learning. Over the course of the internship, participants will immerse themselves in Python's versatile ecosystem, gaining hands-on experience in various domains such as web development, data analysis, machine learning, and automation. Eager to apply for the Python internship opportunity at your esteemed organization, driven by a strong passion for programming and a deep appreciation for Python's versatility and power. With a solid foundation in computer science, honed my skills through coursework and personal projects that emphasize Python's capabilities in data analysis, web development, and automation. Experience includes developing efficient scripts to automate repetitive tasks, creating web applications using frameworks like Django and Flask, and analyzing large datasets with libraries such as Pandas and NumPy. Additionally, a keen interest in machine learning, evidenced by projects where implemented algorithms using TensorFlow and Scikit- learn.Confident that my problem-solving abilities, combined with my enthusiasm for continuous learning and collaboration, make me a valuable addition to your team.Excited about the opportunity to contribute to real-world projects and grow professionally under the guidance of experienced mentors at your company.

# TABLE OF CONTENT

1. About the Company 01
   1. Objective 02
   2. [Reference 02](#_TOC_250015)
   3. [About the Internship 0](#_TOC_250014)3
2. Introduction, Variables, and Data Types 04
   1. [History 05](#_TOC_250013)
   2. [Features](#_TOC_250012)  05
   3. [Installation and Execution 06](#_TOC_250011)
   4. [Hello World! 06](#_TOC_250010)
   5. [Input and Output 07](#_TOC_250009)
   6. [Basic Data Types and Operators 07](#_TOC_250008)
   7. Strings 15
   8. Compound Data Types 08
3. Control Structures 09
   1. [Conditions 09](#_TOC_250007)
   2. [Loops 10](#_TOC_250006)
4. Functions, Modules, and Packages 11-12
5. Files and Regular Expressions 12
   1. File Input/Output 12
   2. [Text Processing 13](#_TOC_250005)
   3. [Pattern Matching and Regular Expressions 14](#_TOC_250004)
   4. [Application: Querying Publication Data 15](#_TOC_250003)
6. Django Framework 16
   1. [Installing and Running Django 17](#_TOC_250002)
   2. [Creating and Running a Web Application](#_TOC_250001) 18
   3. [Parameter Passing with GET 19](#_TOC_250000)
7. Conclusion 20
8. Internship Outcomes 21
9. Criteria for Certification 22
10. Internship Certificate 23
11. Daily Diary 24-25

# ABOUT THE COMPANY



Our company’s CGS primary objective is to create value by providing software and services in IT as well as NON IT that solve specific problems or fulfill particular needs.We are the best providers for IT skill trainings.

We aim to make you develop, market, and work on software to meet the needs of individuals and organizations. This includes designing, building, testing, and maintaining software that can be used for various purposes, including productivity, entertainment, communication, and security.

CSG Internship Programs:

CSG’s internship programs provide a dynamic and immersive experience for aspiring professionals. We believe in nurturing talent and providing a platform for individuals to apply their academic knowledge to real-world scenarios.

Our internships are designed to expose participants to the intricacies of the industry, fostering a collaborative and innovative mindset.

### OBJECTIVE

The primary objective of the Python internship is to provide interns with hands-on experience and practical knowledge in Python programming, fostering their development into proficient and capable software developers. This internship is designed to achieve the following specific goals:

* + - **Skill Development:** Enhance interns' understanding of Python programming, including core concepts, syntax, and advanced features. This includes object-oriented programming, data structures, algorithms, and libraries such as NumPy, Pandas, and Flask/Django for web development.
    - **Practical Experience:** Offer real-world coding experience through involvement in active projects. Interns will work on tasks that mirror the challenges faced in a professional environment, including debugging, code reviews, and version control using Git.
    - **Project Management:** Introduce interns to software development lifecycle processes, including requirement analysis, design, implementation, testing, and deployment. This helps interns understand the importance of planning and documentation.
    - **Portfolio Development:** Enable interns to build a strong portfolio of work, showcasing their skills and accomplishments to potential employers through completed projects and contributions to open-source initiatives.

By the end of the internship, interns will have gained substantial knowledge and experience in Python programming, preparing them for successful careers in software development and related fields.

### Reference

<https://slashmark.cloud/login1.php>

### About the Internship

This internship program offers a comprehensive dive into the world of Python programming, focusing on practical applications and experiential learning. Over the course of the internship, participants will immerse themselves in Python's versatile ecosystem, gaining hands-on experience in various domains such as web development, data analysis, machine learning, and automation. I am eager to apply for the Python internship opportunity at your esteemed organization, driven by a strong passion for programming and a deep appreciation for Python's versatility and power. With a solid foundation in computer science, I have honed my skills

through coursework and personal projects that emphasize Python's capabilities in data analysis, web development, and automation. My experience includes developing efficient scripts to automate repetitive tasks, creating web applications using frameworks like Django and Flask, and analyzing large datasets with libraries such as Pandas and NumPy. Additionally, I have a keen interest in machine learning, evidenced by projects where I implemented algorithms using TensorFlow and Scikit-learn. I am confident that my problem-solving abilities, combined with my enthusiasm for continuous learning and collaboration, make me a valuable addition to your team. I am excited about the opportunity to contribute to real-world projects and grow professionally under the guidance of experienced mentors at your company.

### INTRODUCTION

### History

Python is a widely used programming language today. Starting from the first semester students to final year projects to industry personnel, Python finds its use in developing programs for graphics applications, text processing, data analysis, among others.

Python was developed by Guido van Rossam, a Dutch programmer, and released in 1991. The name is inspired from a BBC comedy show *Monty Python’s Flying Circus*. Python is a successor of the ABC programming language. At the time of this writing, Python 3 is the latest major release of Python (on your computers, you may notice the program *python3*).

Since the last 20 years, Python has been in the Top 10 most popular programming languages. At the time of this writing (July 2022), Python is the most popular language, surpassing C and Java (according to the TIOBE index).



### Features

**Figure 2.1:** “Twosnakes” logo of Python

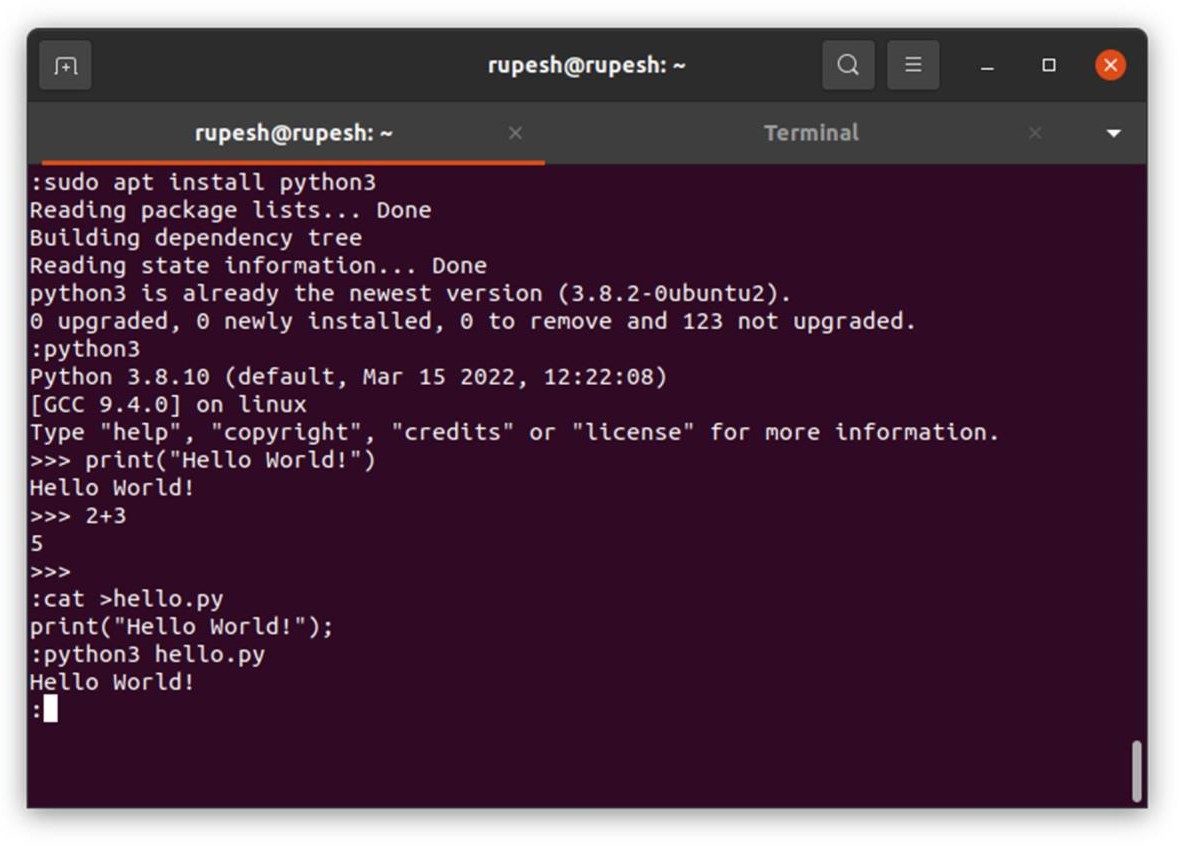
Python is a general-purpose programming language and supports multiple paradigms or ways of programming. For instance, we can write procedural (sequence of steps) as well as object-oriented programs (entities as objects and communication using messages across them) in it. It can also be used to write functional programs (applying and composing functions), among others. Unlike C and C++, Python is interpreted. This means Python programs are not compiled and stored into a binary code file (e.g., an executable)., but its source is translated into machine code and executed by the interpreter directly (without us seeing the executable code). Thus, on your computers, *python3* is an interpreter (and *gcc* is a compiler for C programs).

Python is also dynamically-typed. This means that the type of a variable may not be specified in the source code, and is identified when the program executes. Python also relies on garbage collection which reclaims the allocated memory of the variables which are no longer needed (referenced, to be precise). This relieves the programmer of the task of memory deallocation, similar to Java. Python alsohas a large variety of standard libraries, which allow us to write complex codes quickly, improving ourproductivity.

### Installation and Execution

If you have access to the internet, you can write and execute Python programs online (e.g., Replit, Code Academy, Codevny). If you wish to install it on your laptop or iPad, you can download the appropriate installable for Windows or Linux or iPadOS or other operating systems from Python’s official website [www.python.org.](http://www.python.org/)

For instance, the following screenshot shows a Linux installation and running of Python.



Once installed, you can invoke the Python interpreter (using graphical user interface or via a commandline) to execute a Python program. Python programs can be written in your favorite editor (e.g., VS Code or Sublime or gedit or even notepad) apart from the Python IDE, called IDLE on Windows.

On the command line, you can use certain basic commands to navigate through the file system. For instance, if your home directory is /home/user, you can create a directory for Python programs as:

$ cd # go to home directory

$ mkdir python # create directory

$ cd python # go into that directory

$ cat >hello.py# create a new file hello.py print(“Hello World!”)

$ python3 hello.py # run the Python interpreter

$ cd .. # come back to home directory

Python programs are stored in files typically with extension .py (e.g., *hello.py*). On a command-line, we can execute the program as below.

$ python3 hello.py

In the above command-line, $ indicates the command-prompt, *python3* is the interpreter, and *hello.py* is a text-file containing your program. On your computer, the interpreter name may differ depending upon your installation. For instance, on Windows, the interpreter binary is named *python*. Further, with new versions of Python, the binary name may change to *python4*.

### Hello World!

A typical first program in a programming language prints Hello World! to the screen. Below is our first program in Python.

**print**(”Hello World!”)

The program uses a function *print*() to output a message to the screen. The message is given as an argument to *print*() and is specified in double-quotes (“). It can also be specified in single-quotes (‘) or triple-quotes (‘’’ or “””). When executed using the interpreter, it outputs the message.

$ python3 hello.pyHello World!

The last $ indicates that the command-prompt is displayed again for the next command.

### Input and Output

The function *print*() is used for output, while the function *input*() is used for taking input from the user.

Consider the following functionality where we wish to take the user’s name as input and greet the user.

$ python3 greet.py What is your name?Guido van Rossam

Hello Guido van Rossam

One problem here is that we would like to greet exactly the name that was entered. This demands us to store the name during input and *retrieve* it during the output. Such an entity is called a variable. A

variable is capable of storing a value which can be retrieved later. In fact, a variable can hold different values at different times. Our program achieving the above functionality looks like this.

1. print("What is your name?")
2. name = **input**()
3. print("Hello", name)

### Basic Data Types and Operators

Python supports numeric types such as integer, floating point, as well as complex numbers. It also supports boolean values and strings. Further, the value of a variable can be converted from one type toanother (as we saw in the last example).

**Example**: A card is drawn at random from a deck of well-shuffled cards. Find the probability of it beingneither a king nor a spade.

ncards = int(52) nkings = int(4) nspades = int(13) nspadeking = int(1)

nnonspadenonking = ncards - (nkings + nspades - nspadeking) probnonspadenonking = nnonspadenonking / ncards print('Probability of nonking, nonspade is', probnonspadenonking)

The output of this computation is:

Probability of nonking, nonspade is 0.6923076923076923

We can restrict the output to a few decimal digits using format specification (similar to C).

### 9.7 Strings

String is a fundamental data type in Python and the language provides many ways of manipulating strings. Strings are enclosed in single-, double-, or triple-quotes. Triple-quoted strings can span multiple lines. Certain characters such as ‘\n’ have special meaning and are not treated as two characters, but one. These are called escape characters or escape sequences. Strings can be concatenated readily by using the + operator. Note that concatenation does not add a space between the two strings.

**Example**: Extract fields from a roll number.

Consider an institute with roll numbers of the following format. An example roll number is CS23B010.

* The roll number is exactly eight letters long.
* The first two letters indicate the department (e.g., CS, ME, EE, AE).
* The next two digits indicate the admission year (e.g., 23, 22, 21, 20).
* The next letter is for degree (e.g., B for BTech, M for MTech, S for MS, P for PhD).
* The last three digits indicate the position within the class.

Our task is to extract these different fields from an input roll number. The solution program isgiven below.

rollno = input() branch = rollno[0:2] year = int(rollno[2:4])

# CS23B010

# CS

# 23

degree = rollno[4] # B

position = int(rollno[5:8]) # 010

print(branch, "20%02d" %year, degree, position, sep=',')

### 2.7 Compound Data Types

We now introduce the compound data types such as lists, tuples, and dictionaries. All these are aggregates of several elements (which could themselves be aggregates). We will make use of these later in an elaborate manner.

|  |  |  |  |
| --- | --- | --- | --- |
| Data type | **List** | **Tuple** | **Dictionary** |
| Properties | Ordered, allows duplicates, mutable | Ordered, allows duplicates, immutable | Associative, unique keys, mutable |
| Create | L = [0, 1, 2, “T”, 4] | T = 0, 1, 2, “T”, 4, 5  or  T = (0, 1, 2, “T”, 4, 5) | D={'IN':91,'US':1,‘AU’:7}  or  D = dict(IN=91, US=1, AU=7)  or  D = dict([('IN', 91), ('US',  1), (‘AU’, 7)]) |
| Index | L[0], L[-1] | T[0], T[-1] | D['IN'], D[‘AU’] |
| Range / Slice | L[1:5] | T[1:5] | Not supported |
| Length | len(L) | len(T) | len(D) |
| Concatenation | L + L | T + T | D1 | D2  (Python 3.9 onward) |
| Mutation | L[0] = 10 | Not supported | D['US'] = 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Append | L.append(6) | Not supported | Use mutation |
| Remove | L[2:12] = [] | Not supported | del(D['US']) |
| Unpack | e1, e2, e3 = L | e1, e2, e3 = T | e1, e2, e3 = D.items() |
| Others | Empty list as []  One length list as [1] | Empty tuple as ()  One length tuple as (1,) | Empty dictionary as {} One length dictionary as  {0:’zero’} |

#### SUMMARY

We touched upon the historical aspects of Python’s genesis and delved into writing simple programs using numbers and strings. We also introduced aggregates such as lists, tuples, and dictionaries.

### Conditions

The duplication can be avoided if we can combine the *if* conditions on Lines 2 and 3. Python allows us to do that using conjuncts such as *and* and *or*.

* + 1. rollNum = input("Your roll number: ")
    2. if rollNum[0] == 'C' **and** rollNum[1] == 'S':
    3. print("Hi Bro!")
    4. else:
    5. print("Excuse me?")
    6. The conjunct executes the *if*-block if both the conditions are true. Otherwise, it executes the *else*- block.Line 2 can also be written succinctly as:

7.

8. 2. if rollNum[0:2] == 'CS':

**Example**: Find the student from your department where the roll number may be in capital or small-case letters.

1. rollNum = input("Your roll number: ")
2. if (rollNum[0] == 'C' **or** rollNum[0] == 'c') **and** (rollNum[1] == 'S' **or** rollNum[1]

== 's'):

1. print("Hi Bro!")
2. else:

To address this, we need to augment our *if* condition to include small-case letters too.

Thus, the *or* conjunct evaluates to True if any one of the conditions evaluates to True. That is, it evaluates to False only if all the conditions are False.

Note the use of parentheses to combine the clauses by the *and* conjunct. This is required because *or* hasa

lower precedence than the *and* conjunct. Thus, in absence of parentheses, the meaning of Line 2 wouldbe:

2. if rollNum[0] == 'C' **or** (rollNum[0] == 'c' **and** rollNum[1] == 'S') **or** rollNum[1] == 's':

Can you find out the inputs for which this modified program would produce wrong results?One may write the above program by reordering the conditions.

Are the last two programs equivalent? The answer is no. The last program allows roll numbers ‘CS…’and ‘cs…’, but does not allow a mixed-case ‘Cs…’ or ‘cS…’, which is permitted by the earlier program.

### Loops

Some programs cannot be written without the ability to repeat. For instance, consider printing ‘Hello World!’ 100 times. One can write 100 *print*() statements easily. However, if I now ask you to write a program to take a number from the user and print ‘Hello World!’ those many times, you would be stuck!

Loops allow us to repeat an arbitrary piece of code, arbitrarily number of times. Python supports two types of loops: *while* and *for*. The *while* loop iterates through (repeats) a sequence of code till a given condition is True. The *for* loop iterates over the items of a givensequence. We will study both in detail now.

#### While Loop

Let’s first write our program to print a message a certain user-defined number of times.

1. n = int(input())2.

i = 0

1. **while** i < n:
2. print('Hello World!')

Note the similarity of the loop’s structure with that of an *if* statement. The body of the loop (in this case, Line 4) is repeated till the condition on Line 3 is true.

If you enter 10 as input, how many ‘Hello World!’s does our program print? 9 or 10? Well, it continues to print ‘Hello World!’ an unbounded number of times. Why? Because we asked it to. The condition *i < n* continues to remain true, as the value of *i* never changes in the loop. To get the expected result, we need to increment *i*.

1. n = int(input()) 2. i = 0

1. **while** i < n:
2. print('Hello World!') 5. i = i + 1

# progress

#### SUMMARY

We explored various basic control constructs in Python such as conditionals and loops. We solved various problems using those constructs. We also looked at various loop modifiers which allow special control flow.

### 4.Functions

Functions allow us to separate a specific functionality and reuse it from multiple places. For instance, we use the functions *print* and *len*() which are defined in the standard Python library. But we can writeour own functions too!

Let’s begin with defining a function for printing a message and calling it in different ways.

1.

2.

3.

4.

5.

6.

7.

8.

9.

10.

**def hello**():

print("Hello World!")

# function definition

**hello**()

# function call

if False:

**hello**()

# function call inside a conditional

for i in range(1, 3):

**hello**()

# function call inside a loop

**Modules**

Modules are what we have been writing so far. Each program we have created is a module, just that we did not look at them that way. Typically, a module is a collection of cohesive functions. For instance, while designing a large application, we may want to keep all the functions related to databaseprocessing, frontend, error handling, utilities in different modules. Let’s recall our prime number program. Say, we have stored it in a file ***primefun****.py*.

1.

2.

3.

4.

5.

6.

7.

8.

9.

10.

11.

12.

import math

# name of this file is **primefun**.py

def **isPrime**(n): if n % 2 == 0: return False

for i in range(3, 1 + int(math.sqrt(n)), 2): if n % i == 0:

return False

else:

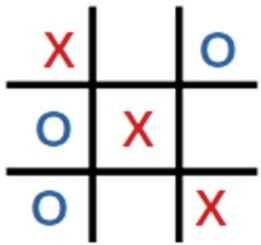
return True

1. for n in range(3, 100+1):
2. if **isPrime**(n): print(n, end=', ')
3. print()

### Packages

A package encapsulates multiple modules of a project. From a data organization perspective, a module is stored in a .py file, while a package is the directory containing the .py file. We can then access a module *mod* in package *pack* as *pack.mod*. In general, a module file may not be directly inside a packagedirectory, but may be nested deeper, due to multiple subdirectories – for instance, pack.subdir1.subdir2.mod. In such cases, each of subdir1, subdir2 etc. are called subpackages.

Let’s consider a concrete example of a project. Say, we wish to implement a simple game – tic-tac-toe. A sample is shown below. In a 3x3 grid, two players fill in two symbols (say, X and O) in empty cells alternatively, and the first person getting three symbols in a row or a column or a diagonal wins. Sometimes, the game may end in a draw.



The game can be nicely implemented as a package, since it has multiple components. We need to havethe game logic in one module, and game display in another. We can add more functionality such as storing the game in a file or storing the scores of the players which can be loaded from a database etc. But for now, we will constrain ourselves with two subpackages: *logic* and *display*. Within each of these subpackages, we may have multiple modules; for instance, text-based display and GUI-based display. The

overall directory structure looks like this:

**tictactoe**/ # package

├── init .py # initializer

├── **display** # subpackage

│ ├── init .py # initializer

│ ├── gui.py # GUI module

│ └── text.py # text module

├── **logic** # subpackage

│ ├── init .py # initializer

│ └── twoplayer.py # module

└── internal # regular directory, not a subpackage

The outermost directory represents the main package. Thus, if we have a user program using this package,

import tictactoe

print("in ttt")

tictactoe.**loadGame**("text") # argument decides whether to load the game in text or GUI.

say *ttt.py*, it will have the code as below

#### SUMMARY

We learned various ways to group statements using functions, functions using modules, and modules using packages. In the process, we developed slightly larger codes which allow us to develop applications. **5.1File Input/Output**

In a Linux terminal, the command to print the contents of a file is *cat*, which can be used as follows:

$ ls hello.txt

$ cat hello.txt This is a text file.

It contains four lines.

# In Windows terminal: *dir*

# In Windows terminal: *type*

The third line is empty.

Let’s implement this functionality.

**Example**: Print contents of a file.

1. myfile = **open**('hello.txt') 2.

3.

# open the file for reading.

4.

5.

6.

for line in myfile: print(line)

myfile.**close**()

# close the file.

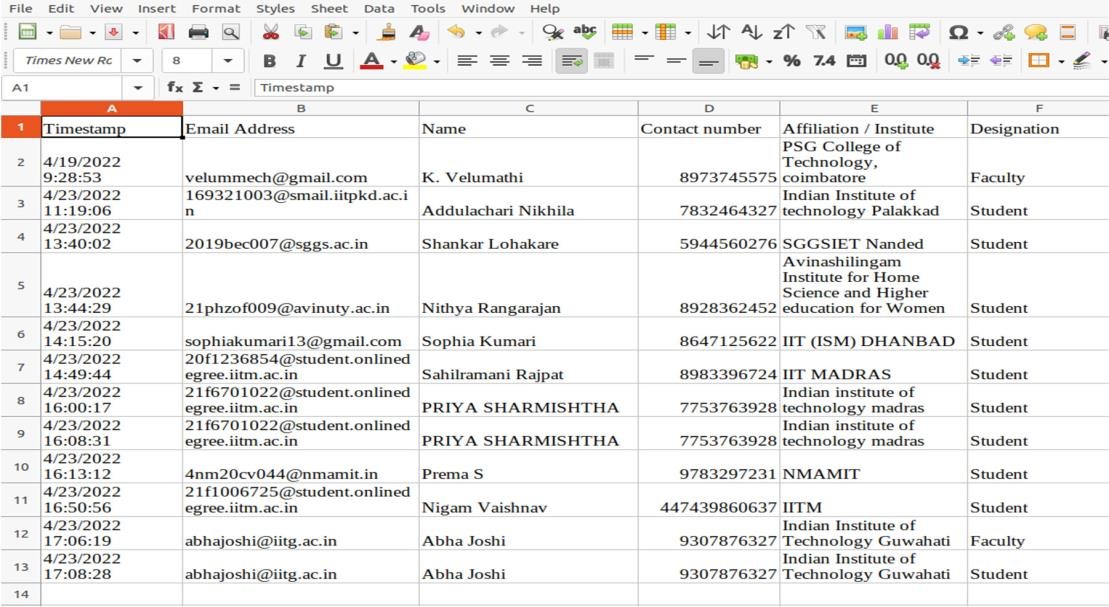
A file is opened for processing in Line 1 (the operating system sets up resources using which the file can be accessed), processed (Line 3), and closed in Line 6 (allowing the operating system to release the resources).

### Text Processing

Python is well-suited for processing text data, among other aspects. First, string is a basic type supported. Second, there are several functions provided for filtering and manipulating strings. We will explore many of these functions via developing an application’s backend.

**Example**: Process conference registration data.

Consider that you have developed a web-form (say, using Google Forms) which collects information of registrants and stores it in a spreadsheet. The basic information supplied is name, email, phone, affiliation, and designation, and the form also captures the time of registration.

There are libraries which support processing of spreadsheets. In our application, we can convert it intoa text-based .csv (comma-separated values) file, so that it can be processed as a tabular data.

### Pattern Matching and Regular Expressions

Sometimes, an exact match cannot be specified to cover all the cases. Consider, for instance, finding a mobile number from a text. One may find it easy to go over the text and identify ten consecutive digits. This is alright, but sometimes the mobile number is written in different forms, such as 99406 67821 (with a space or a hyphen), while at times as (994) 066-7821 (e.g., in the US). One can write a programto identify all such patterns. Regular expressions allow us to specify those patterns succinctly.

Python provides a module *re* for regular expressions. We will start with a function *findall*() to identify all occurrences of a given pattern in a text.

**Example**: Find all the mobile numbers out of a long text.We will first show the program and then explain.

1.

2.

3.

4.

import re

text = """

Hello, I am Dr. Mobile123. Please call me in case of an emergency. My phone number is 99405 33241. Otherwise, you can call my assistant at 8932732436. I am available at my clinic from 9 to 1. For appointments, call the reception at (687) 324-3232.

"""

5.

6.

7.

8.

9.

digits10 = r'\d{10}' digits55 = r'\d{5}\D\d{5}'

10. digitsus = r'\(\d{3}\)\s?\d{3}\D\d{4}'

# 10 consecutive digits # 5 digits gap 5 digits

# (3 digits) 3 digits - 4 digits

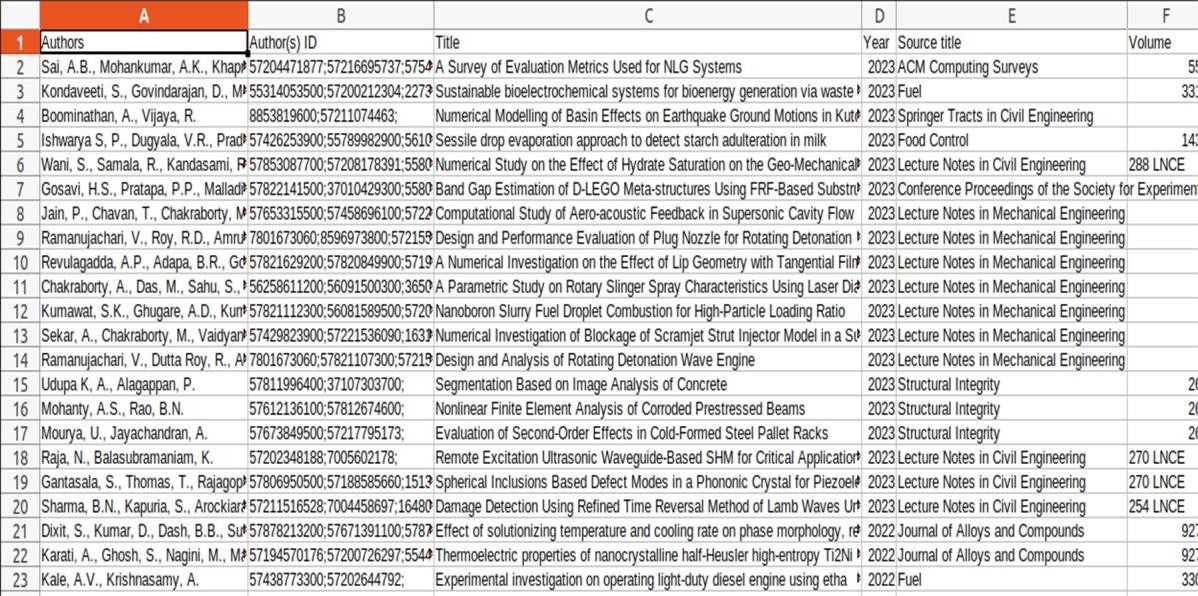
1. regex = digits10 + '|' + digits55 + '|' + digitsus # first or second or third pattern
2. mobiles = re.**findall**(regex, text)
3. print(mobiles)

The output of the above program is:

['99405 33241', '8932732436', '(687) 324-3232']

#### Application: Querying Publication Data

We put all our learning together to create a real-world application. We are given publication data fromScopus, and we would like to query it to identify various records of interest. Snapshot of the data is given below. Its first line shows various columns such as Authors, Author(s) ID, Title, Year, etc.



By now, we know how to convert this data into a .csv file and read the records in various lists. Note that all the authors are part of the same column. Since author names can be ambiguousor written differently in different documents (e.g., R. Nasre or Rupesh Nasre), Scopus maintains a unique ID per author, which is given in the second column. We would like to query this information. All the code along with sample data is available on the course webpage2. We note that there is a difference in the way Windows and Linux store newline characters in files. Therefore, your program may not directly work across the platforms. To ease out your processing, the course webpage lists files in both ‘dos’ (for Windows) and ‘unix’ (for Linux) formats. Also, to handle special characters in Windows, one may need to change the encodingto UTF-8 as follows.

df = open(datafile, **encoding=’utf8’**)

#### SUMMARY

We learned to read and write to files, and to process those depending upon the application. We then digged deeper into various string related functions provided by the language. Finally, we explored the powerful world of regular expressions, allowing us to go beyond a simple *find*(). In each of these, we solved several problems.

### Installing and Running Django

If Django is not already installed, it can be readily installed with the usual commands. For instance, on Linux, the following command can be used.

sudo apt install python3-django

On Windows, it is a two step process. In the first, we create a virtual environment, and then we can install Django via pip installer, as follows. All these commands can be executed either in *cmd* command prompt in Windows or in its *PowerShell*.

python -m venv myvenv myvenv\Scripts\activate python -m pip install Django

The first command creates a virtual environment in the directory *myvenv*. The second commandactivates it. The third command uses *pip* to install the Django framework.

To check if Django is already installed or whether it got installed correctly, try running the followingcommand from your home directory (a directory you have write access to).

django-admin startproject mydjango

Here, *mydjango* is a project name of your choice. If the command runs successfully, it would not show

$ ls -R mydjango/

mydjango/:

db.sqlite3 manage.py mydjango

mydjango/mydjango:

asgi.py init .py pycache settings.py urls.py wsgi.py

mydjango/mydjango/ pycache :

init .cpython-38.pyc settings.cpython-38.pyc urls.cpython-38.pyc wsgi.cpython-38.pyc

any output, but will create the following directory structure.

For now, ignore the use of various files and subdirectories. We will get back to them as we create our first web application. But these files should indicate that Django is installed properly. We can now startthe server.

### Creating and Running a Web Application

We are now ready to create our first web application. Navigate to the *mydjango* directory (where

*manage.py* exists) and issue the following command.

$ python3 manage.py startapp appone

This is our first web application, named *appone*. You will see the corresponding directory created in the current directory, with the following structure.

$ ls -R

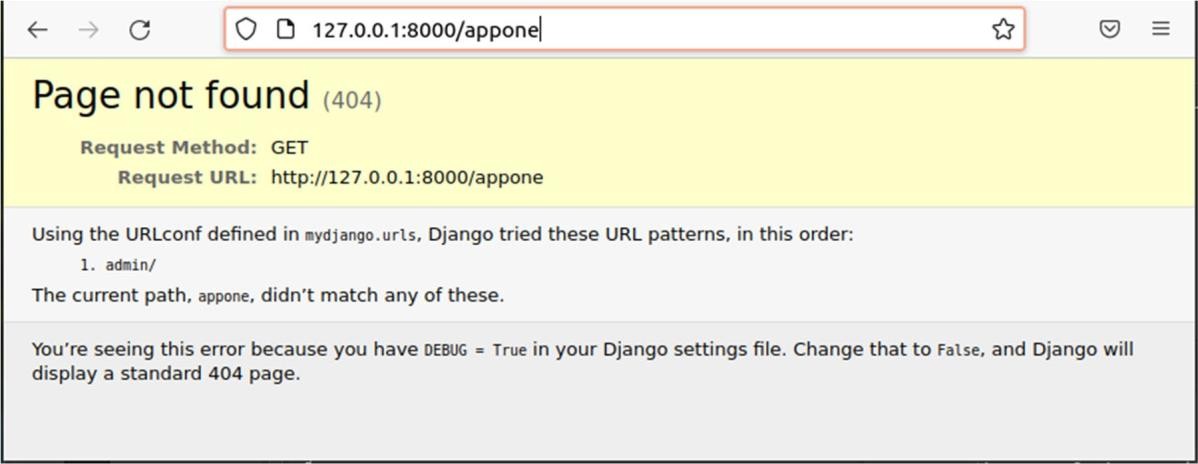
Let’s see if this is sufficient to run our webapp *appone*. We restart the server.

$ python3 manage.py runserver

and in our web-browser, we enter the same url: http://127.0.0.1:8000/

This works as before (with the rocket sign). This means we have not broken anything (so far). Now we check *appone* by entering the url in the browser as: http://127.0.0.1:8000/appone

This shows the browser screen as below.



### Parameter Passing with GET

http://127.0.0.1:8000/expo/?x=2&y=8

Users can pass parameters to our web-app. For instance, we can write an app for exponentiation (it is an overkill, but this is an example). One can then invoke the program as:

1. def **index**(request):
2. x = int(request.**GET**['x'])
3. y = int(request.**GET**['y'])
4. return HttpResponse(str(x\*\*y))

The parameters are named, are assigned values using = operator, and are separated by &. In *expo/views.py*, we can then extract the values of *x* and *y*, compute *x\*\*y* and print the value inthe HTTP response. Django provides a dictionary *GET* within its HTTP request to access these variables.

256

The output shown on the browser will be

You can experiment with other values. Let’s make use of such parameter passing functionalityto create our Django webpage.

**Example**: Create a personal webpage.

Our webpage should have the following display elements.

* + - A common header containing a photo and basic affiliation information.
    - A common menu to navigate through Home, Academics, Projects, and Contacts.
    - When we click on a particular menu item, appropriate information should be displayedbelow the menu.

Sample screenshots of our personal webpage are shown below. The first screen shows the landing Home page.

### CONCLUSION

conclude that, this internship has been a very useful experience for me. safely say that my understanding of the job environment has increased greatly. However, There are some aspects of the job that better and that to work on. To build more confidence in applying accounting principles realized that completed the work earlier. Also, the technical parts of the job were a bit flawed and I was asked multiple times to correct it.

The two main things that I learned after my experience in this firm are the importance of time management and being self-motivated.

### INTERNSHIP OUTCOMES

This internship provides a student with paid professional work experience in a safe and structured environment with help from experts. Typically, an intern will be assigned a worksite mentor and school-based internship coordinator. The worksite mentor will help train a student and advise him or her on how to navigate a particular worksite culture and interact with other workers. The school internship coordinator helps the mentor and student build a strong relationship that will help the intern have positive outcomes from their internship experience. The internship coordinator can also help the student learn how to manage their expectations and implement successful work habits.

Internships help students master professional soft skills such as communication, punctuality. Internships help students master professional soft skills such as communication, punctuality and time management. These are skills that are key for success at a job and college and are highly sought after by companies. Many employers complain that there are few candidates with excellent soft skills. At an internship, a student can practice and improve their industry skills while also learning how to work. Students can gain a better understanding of how what they are learning in school can help them with their future.

### CRITERIA FOR CERTIFICATION

This internship certificate is very important for every undergraduate students who is pursing degree like B-Tech or any other. It is also important for the students of postgraduation. This certificate will gives the best value and weightage for our resume. This internship certificate plays an vital role in our life .

On the off chance that you are interested about work fields and want more experience in field without having to go digging into its academic aspects, opting for an internship to get better understanding about the subject and get closer to the your goals. Internships are programs for students who’ve completed their graduation that give you more work experience and last for a short duration of time. Almost all courses that are taught professionally offer internship programs so the students can get a better sense of the subject and more knowledge about its technicalities before they choose to pursue courses like BBA, B-Tech and MBA.

Having the biggest organizations and managements offering rewarding chances for training to promising experts of each field, internships have become quite popular among students getting their Bachelor’s Degree. With an increasing demand in work experience when students go in for job interviews, internships provide you with a certificate which shows your experience in that particular field and is proved to be beneficial in getting a job. Mentioned below are some advantages of getting additional internship certificates and the effect they have in boosting your career.

Certification Requirements means the requirements specified in this Agreement (including without limitation the Xbox Guide) for quality, compatibility, and/or performance of a Software Title, and, to the extent not inconsistent with the foregoing standards, the standards of quality and performance generally accepted in the console game M industry. Internship certificates are given to those interns who have completed the training at a company or at any institution. This certificate contains proof of internship duration (in months or year), stipends (if any) and other job-related details.

### INTERNSHIP CERTIFICATE



+

### Department of Electronics and Communication Engineering Internship Diary

**Name of the Student:** K.Chaithanya prabha **Roll No:** 20F11A04B9

**Name of the Industry/Organization: Chatrapathi Greentech Softwares PVT.LTD**

**Name of the Track:** Python

|  |  |  |
| --- | --- | --- |
| **S.NO** | **DATE** | **LEARNING OUTCOME** |
| 1 | 1-02-2024 | Self-introduction,- Introduction to Python |
| 2 | 2-02-2024 | Basic python programs, Hands on session |
| 3 | 5-02-2024 | History about Python |
| 4 | 6-02-2024 | Hands on |
| 5 | 7-02-2024 | Features of Python |
| 6 | 8-02-2024 | Installation of python software |
| 7 | 9-02-2024 | Execution of simple program |
| 8 | 12-02-2024 | Hello World |
| 9 | 13-02-2024 | Input and Output |
| 10 | 14-02-2024 | Basic Data Types |
| 11 | 15-02-2024 | Operators |
| 12 | 16-02-2024 | Strings |
| 13 | 19-02-2024 | Compound Data Types |
| 14 | 20-02-2024 | Assignment 1 |
| 15 | 21-02-2024 | Introduction to Control Structures |
| 16 | 22-02-2024 | Conditions |
| 17 | 23-02-2024 | Loops |
| 18 | 26-02-2024 | Introduction to Functions |
| 19 | 27-02-2024 | Absent(Due to Review) |
| 20 | 28-02-2024 | Introduction to modules |
| 21 | 29-02-2024 | Introduction to packages |
| 22 | 01-03-2024 | Hands on |
| 23 | 04-03-2024 | File input/output |
| 24 | 05-03-2024 | Text processing |
| 25 | 06-03-2024 | Pattern Matching and Regular expression |
| 26 | 07-03-2024 | Hands on |
| 27 | 11-03-2024 | Application: Querying Publication Data |
| 28 | 12-03-2024 | Introduction to Django |
| 29 | 13-03-2024 | Installing Django Software | |
| 30 | 14-03-2024 | Absent(Due to Campus Drive) | |
| 31 | 15-03-2024 | Running on Django | |
| 32 | 18-03-2024 | Creating Django application | |
| 33 | 19-03-2024 | Running the Web Application | |
| 34 | 20-03-2024 | Parameter passing with GET | |
| 35 | 21-03-2024 | Introduction to Visual Studio | |
| 36 | 22-03-2024 | Assignment-2 | |
| 37 | 26-03-2024 | Introduction to OOPS | |
| 38 | 27-03-2024 | Foundations of OOPS | |
| 39 | 28-03-2024 | OOPS: Properties | |
| 40 | 01-04-2024 | OOPS Methods | |
| 41 | 02-04-2024 | OOPS Fundamentals: Inheritance | |
| 42 | 03-04-2024 | Exceptions | |
| 43 | 04-04-2024 | Introduction to MISCELLANEOUS | |
| 44 | 05-04-2024 | Generators, Iterators, and Closure | |
| 45 | 08-04-2024 | Files | |
| 46 | 11-04-2024 | OS Module | |
| 47 | 12-04-2024 | Interacting with OS module | |
| 48 | 15-04-2024 | Data type Module | |
| 49 | 16-04-2024 | Calendar Module | |
| 50 | 18-04-2024 | Simple To-Do List Application | |
| 51 | 19-04-2024 | Number Guessing Game | |
| 52 | 22-04-2024 | Python Password Generator | |
| 53 | 24-04-2024 | Chatbot Development | |
| 54 | 26-04-2024 | Python Voice Assistant | |
| 55 | 29-04-2024 | Real-tome Auto License Plate Recognition with Jetson Nano | |
| 56 | 30-04-2024 | Review about the Internship | |

### GUIDE HOD ECE