Third Year B.C.A. (Under Science) Semester VI

Course Code: BCA601 **Course Title: Android Programming**

Total Contact Hours: 48 hrs. **Total Credits: 04** **Total Marks: 100**

(60 Lectures)

Teaching Scheme: Theory- 05 Lect./ Week

Course Objectives:

The objective of this course is to understand the Android Operating System and develop applications using Google's Android open-source platform.

UNIT NO.	DESCRIPTION	No. of LECTURES
UNIT 1	**1. Introduction to Android** 1.1. Overview 1.2. History 1.3. Features of Android 1.4. Architecture of Android • Overview of Stack • Linux Kernel • Native Libraries • Android Runtime • Application Framework • Applications 1.5. SDK Overview • Platforms • Tools – (JDK, SDK, Eclipse/Android Studio, ADT, AVD, Android Emulator) • Versions 1.6. Creating your first Android Application	**06**
UNIT 2	**2. Activities, Fragments and Intents** 2.1. Introduction to Activities 2.2. Activity Lifecycle 2.3. Introduction to Intents 2.4. Linking Activities using Intents 2.5. Calling built-in applications using Intents 2.6. Introduction to Fragments 2.7. Adding Fragments Dynamically 2.8. Lifecycle of Fragment 2.9. Interaction between Fragments	**09**
UNIT 3	**3. Android User Interface** 3.1. Understanding the components of a screen • Views and ViewGroups • LinearLayout • AbsoluteLayout	**10**

	• TableLayout • RelativeLayout • FrameLayout • ScrollLayout • ScrollView 3.2. Adapting to Display Orientation • Anchoring Views • Resizing and Repositioning 3.3. Managing Changes to Screen Orientation • Persisting State Information during Changes in Configuration • Detecting Orientation Changes • Controlling the Orientation of the Activity 3.4. Utilizing Action Bar • Adding Action Items to the Action Bar • Customizing the Action Items and Application Icon	
UNIT 4	**4. Designing Your User Interface with Views** 4.1. Using Basic Views • TextView • Button, ImageButton, EditText, CheckBox • ToggleButton, RadioButton, and RadioGroup Views • ProgressBar View • AutoCompleteTextView View 4.2. Using Picker Views • TimePicker View • DatePicker View 4.3. Using List Views to Display Long Lists • ListView View • Using the Spinner View 4.4. Understanding Specialized Fragments • Using a ListFragment • Using a DialogFragment • Using a PreferenceFragment	**10**
UNIT 5	**5. Displaying Pictures and Menus** 5.1. Using Image Views to Display Pictures • Gallery and ImageView views • Image Switcher • Grid View 5.2. Using Menus with Views • Creating the helper methods • Options Menu • Context Menu	**05**
UNIT 6	**6. Databases – SQLite** 6.1. Introduction to SQLite 6.2. SQLiteOpenHelper and SQLiteDatabase 6.3. Creating , opening and closing database 6.4. Working with cursors, Insert, Update, Delete 6.5. Building and executing queries	**06**

UNIT 7	**7. Messaging and E-mail** 7.1. SMS Messaging • Sending SMS Messages Programmatically • Getting Feedback after Sending a Message • Sending SMS Messages Using Intent • Receiving SMS Messages • Caveats and Warnings 7.2. Sending E-mail	**06**
UNIT 8	**8. Location-Based Services and Google Map** 8.1. Display Google Maps • Creating the project • Obtaining the Maps API Key • Displaying the Map • Displaying the Zoom Control • Changing Views • Navigating to a specific location • Adding Markers • Getting the location that was touched • Geocoding and Reverse Geocoding 8.2. Getting Location Data 8.3. Monitoring a Location	**08**

Reference Books:

1. Beginning Android4 Application Development, By Wei-Meng Lee WILEY India Edition WROX Publication
2. Professional Android 4 Application Development, By Reto Meier WROX Publication
3. The official site for ***Android developers* -** https://developer.android.com

Third Year B.C.A. (Under Science) Semester VI

Course Code: BCA602 **Course Title: Python Programming**

Total Contact Hours: 48 hrs. (60 Lectures) **Total Credits: 04** **Total Marks: 100**

Teaching Scheme: Theory- 05 Lect./ Week

Course Objectives:

- To introduce various concepts of programming to the students using Python.
- Students should be able to apply the problem solving skills using Python

Unit No.	Contents	No. of Lectures
Unit 1	**Introduction to Python Scripting** • Why Scripting is Useful in Computational Science • Classification of Programming Languages • Productive Pairs of Programming Languages • Gluing Existing Applications • Scripting Yields Shorter Code, Efficiency • Type-Specification (Declaration) of Variables • Flexible Function Interfaces • Interactive Computing • Creating Code at Run Time • Nested Heterogeneous Data Structures • GUI Programming • Mixed Language Programming • When to Choose a Dynamically Typed Language • Why Python? Script or Program? • Application of Python • Concept (immutable)	**04**
Unit 2	**Basic Python** • Python identifiers and reserved words • Lines and indentation, multi-line statements • Comments • Input/output with print and input functions, • Command line arguments and processing command line arguments • Standard data types - basic, none, Boolean (true & False), numbers • Python strings • Data type conversion • Python basic operators (Arithmetic, comparison, assignment, bitwise logical) • Python membership operators (in & not in) • Python identity operators (is & is not) • Operator precedence • Control Statements, Python loops, Iterating by	**06**

	subsequence index, loop control statements (break, continue, pass) • Mathematical functions and constants (import math), Random number functions	
Unit 3	**Python strings** • Concept, escape characters • String special operations • String formatting operator • Single quotes, Double quotes, Triple quotes • Raw String, Unicode strings, Built-in String methods. • Python Lists - concept, creating and accessing elements, updating & deleting lists, basic list operations, reverse • Indexing, slicing and Matrices • built-in List functions • Functional programming tools - filter(), map(), and reduce() • Using Lists as stacks and Queues, List comprehensions	**06**
Unit 4	**Python tuples and sets** • Creating & deleting tuples • Accessing values in a tuple • Updating tuples, delete tuple elements • Basic tuple operations • Indexing, slicing and Matrices, built- in tuple functions. • Sets - Concept, operations.	**06**
Unit 5	**Python Dictionary** • Concept (mutable) • Creating and accessing values in a dictionary • Updating dictionary, delete dictionary elements • Properties of dictionary keys • built-in dictionary functions and methods.	**04**
Unit 6	**Functions** • Defining a function (def) • Calling a function • Function arguments - Pass by value, Keyword Arguments, default arguments • Scope of variable - basic rules • Documentation Strings • Variable Number of Arguments • Call by Reference • Order of arguments (positional, extra & keyword) • Anonymous functions • Recursion • Treatment of Input and Output Arguments • Unpacking argument lists • Lambda forms • Function Objects • function ducktyping & polymorphism	**08**

	• Generators (functions and expressions) and iterators, list comprehensions	
Unit 7	**Files and Directories** • Creating files • Operations on files (open, close, read, write) • File object attributes, file positions, Listing Files in a Directory • Testing File Types • Removing Files and Directories • Copying and Renaming Files • Splitting Pathnames • Creating and Moving to Directories • Traversing Directory Trees • Illustrative programs: word count, copy file	**06**
Unit 8	**Python Classes / Objects** • Object oriented programming and classes in Python - creating classes, instance objects, accessing members • Data hiding (the double underscore prefix) • Built-in class attributes • Garbage collection : the constructor • Overloading methods and operators • Inheritance - implementing a subclass, overriding methods • Recursive calls to methods • Class variables, class methods, and static methods	**08**
Unit 9	**Python Exceptions** • Exception handling : assert statement • Except clause - with no exceptions and multiple exceptions • Try - finally, raising exceptions, user-defined exceptions.	**02**

<u>Reference Books:</u>

1. Introducing Python- Modern Computing in Simple Packages – Bill Lubanovic, O'Reilly Publication
2. Beginning Python: From Novice to Professional, Magnus Lie Hetland, Apress
3. Practical Programming: An Introduction to Computer Science Using Python 3, Paul Gries, et al., Pragmatic Bookshelf, 2/E 2014
4. Introduction to Computer Science Using Python- Charles Dierbach, Wiley Publication Learning with Python ", Green Tea Press, 2002
5. E-Books : python_tutorial. pdf, python_book_01.pdf
6. Beginning Programming with Python for Dummies Paperback – 2015 by John Paul Mueller
7. A Beginner's Python Tutorial: http://en.wikibooks.org/wiki/A Beginner%27s Python Tutorial.

Third Year B.C.A. (Under Science) Semester VI

Course Code: BCA603 **Course Title: Recent Trends in IT (Internet of Things)**

Total Contact Hours: 48 hrs. (60 Lectures) **Total Credits: 04** **Total Marks: 100**

Teaching Scheme: Theory- 05 Lect./ Week

Pre-Requisite: Basic understanding of electronics and microprocessors.

Course Objectives:

1. The Internet of Things (IoT) is aimed at enabling the interconnection and integration of the physical world and the cyber space.
2. To learn about SoC architectures, programming Raspberry Pi and implementation of internet of things and protocols.

Expected Learning Outcomes:

1. Enable learners to understand System On Chip Architectures.
2. Introduction and preparing Raspberry Pi with hardware and installation.
3. Learn physical interfaces and electronics of Raspberry Pi and program them using practical's
4. Learn how to design IoT based prototypes.

Unit No.	Contents	No. of Lecctures
Unit 1	**System on Chip (SoC) and Internet of Things (IoT) Overview** - **System on Chip:** What is System on chip? Structure of System on Chip. - **SoC products:** Field Programmable Gate Array **(**FPGA)**,** General Purpose Graphics Processing Units **(**GPU), Accelerated Processing Unit (APU), Compute Units. -**The IoT paradigm** giving overview of IoT supported Hardware platforms such as: Raspberry pi, SoC on ARM 8 Processors, Arduino and Intel Galileo boards. -**Network Fundamentals**: Wired Networking(Router, Switches), Wireless Networking(Access Points) -**Introduction to Raspberry Pi:** Introduction to Raspberry Pi, Raspberry Pi Hardware, Preparing your raspberry Pi. -**Raspberry Pi Boot:** Learn how this small SoC boots without BIOS. Configuring boot sequences and hardware. -**Introduction to IoT:** What is IoT? IoT examples, Simple IoT LED Program. -**IoT and Protocols** -**IoT Security:** HTTP, UPnp, CoAP, MQTT, XMPP**.** -**IoT Service as a Platform:** Clayster, Thinger.io, SenseIoT, carriots and Node RED. -**IoT Security and Interoperability:** Risks, Modes of Attacks, Tools for Security and Interoperability.	**20**

Unit 2	**Programming Raspberry Pi** **Raspberry Pi and Linux:** About Raspbian, Linux Commands, Configuring Raspberry Pi with Linux Commands **Programing interfaces:** Introduction to Node.js, Python. **Raspberry Pi Interfaces:** UART, GPIO, I2C, SPI **Useful Implementations:** Cross Compilation, Pulse Width Modulation, SPI for Camera.	**15**
Unit 3	**Case Study & advanced IoT Applications:** IoT applications in home, infrastructures, buildings, security, Industries, Home appliances, other IoT electronic equipments. Use of Big Data and Visualization in IoT, Industry 4.0 concepts. Sensors and sensor Node and interfacing using any Embedded target boards (Raspberry Pi / Intel Galileo/ARM Cortex/ Arduino)	**15**
Unit 4	**Internet of Things Privacy, Security and Governance** Introduction, Overview of Governance, Privacy and Security Issues, Contribution from FP7 Projects, Security, Privacy and Trust in IoT-Data-Platforms for Smart Cities, First Steps Towards a Secure Platform, Smartie Approach. Data Aggregation for the IoT in Smart Cities, Security	**10**

TEXT BOOKS:

1. 6LoWPAN: The Wireless Embedded Internet, Zach Shelby, Carsten Bormann, Wiley
2. Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems, Dr. Ovidiu Vermesan, Dr. Peter Friess, River Publishers
3. Interconnecting Smart Objects with IP: The Next Internet, Jean-Philippe Vasseur, Adam Dunkels, Morgan Kuffmann
4. Internet of Things : A hands- on Approach by Arshdeep Bahga, Vijay Madisetti
5. IoT Programming: A Simple and Fast Way of Learning IOT by David Etter

REFERENCES:

1. The Internet of Things: From RFID to the Next-Generation Pervasive Networked Lu Yan, Yan Zhang, Laurence T. Yang, Huansheng Ning
2. Internet of Things (A Hands-on-Approach) , Vijay Madisetti , Arshdeep Bahga
3. Designing the Internet of Things , Adrian McEwen (Author), Hakim Cassimally
4. “Mobile Computing,” Tata McGraw Hill, Asoke K Talukder and Roopa R Yavagal, 2010.
5. Computer Networks; By: Tanenbaum, Andrew S; Pearson Education Pte. Ltd., Delhi, 4th Edition
6. Data and Computer Communications; By: Stallings, William; Pearson Education Pte. Ltd., Delhi, 6th Edition
7. “Fundamentals of Mobile and Pervasive Computing,” F. Adelstein and S.K.S. Gupta, McGraw Hill, 2009.
8. Cloud Computing Bible, Barrie Sosinsky, Wiley-India, 2010

Third Year B.C.A. (Under Science) Semester V or VI

Course Code: BCA-604 **Course Title: Data Analytics**

Total Contact Hours: 48 hrs. (60 Lectures) **Total Credits: 04** **Total Marks: 100**

Teaching Scheme: Theory- 05 Lect./ Week

Course Objectives:

1. Able to apply fundamental algorithmic ideas to process data.
2. Learn to apply hypotheses and data into actionable Predictions.

Unit No.	Contents	No. of Lectures
Unit 1	• Introduction to data Science ○ Basics of Data ○ What is Data Science? ○ Data science process ○ Stages in data science project • Basics of Data Analytics • Types of Analytics – Descriptive, Predictive, Prescriptive • Statistical Inference ○ Populations and samples ○ Statistical modeling, ○ Probability ○ Distribution ○ Correlation ○ Regression	10
Unit 2	Introduction to Machine Learning • Basics of Machine Leaning • Supervised Machine Learning ▪ K- Nearest-Neighbors, ▪ Naïve Bayes ▪ Decision tree ▪ Support Vector Machines • Unsupervised Machine Learning ▪ Cluster analysis ▪ K means ▪ Association Rule Mining • Apriori algorithms • Regression Analysis ▪ Linear Regression ▪ Nonlinear Regression	25
Unit 3	Data Analytics with Python Programming Numpy ○ Arrays ○ Array indexing ○ Datatypes ○ Array math ○ Broadcasting	15

	SciPy ○ Image operations ○ Distance between points Data analysis and manipulation using Pandas package ○ Importing Data , Creating A DataFrame, ○ DataFrame Methods, ○ Indexing DataFrames, Boolean Indexing ○ Indexing Using Labels , Multi-Indexing ○ Merge DataFrames ○ Sorting DataFrames ○ Apply Function ○ Pivot Table, Crosstab ○ Iterating over rows of a dataframe	
Unit 4	Data Visualization ▪ Basic principles, ▪ Ideas and tools for data visualization ▪ Graph Visualization, ▪ Data Summaries, ▪ Model Checking & Comparison ▪ Purpose of visualization ▪ Multidimensional visualization ▪ Tree visualization ▪ Graph visualization ▪ Visualization techniques ▪ Understanding analytics output and their usage ▪ Scikit package ▪ matplotlib library ○ Plotting ○ Subplots ○ Images	10

Reference Books:

1. The elements of statistical learning. Hastie, Trevor, et al., Vol. 2. No. 1. New York: springer, 2009.
2. Applied statistics and probability for engineers. Montgomery, Douglas C., and George C. Runger. John Wiley & Sons,2010
3. Scaling up Machine Learning to White “Hadoop: The Definitive Guide” Third Edition, Bekkerman et al., O‟reilly Media, 2012.
4. “Mining of Massive Datasets”, Anand Rajaraman and Jeffrey David Ullman, Cambridge University Press, 2012.
5. Developing Analytic Talent: Becoming a Data Scientist, Vincent Granville, wiley, 2014.
6. Introduction to Data Science, Jeffrey Stanton & Robert De Graaf, Version 2.0, 2013.
7. “Practical Data Science with R”, Nina Zumel, John Mount, Manning Publications, 2014.
8. “Mining of Massive Datasets”, Jure Leskovec, Anand Rajaraman, Jeffrey D.Ullman, Cambridge

University Press, 2014.

9. "Beginning R - The Statistical Programming Language", Mark Gardener, John Wiley & Sons, Inc., 2012.
10. "An Introduction to R", W. N. Venables, D. M. Smith and the R Core Team, 2013.
11. "Practical Data Science Cookbook", Tony Ojeda, Sean Patrick Murphy, Benjamin Beng fort, Abhijit Dasgupta, Packt Publishing Ltd., 2014.
12. "Visualize This: The Flowing Data Guide to Design, Visualization, and Statistics", Nathan Yau, Wiley, 2011.
13. "Professional Hadoop Solutions", Boris lublinsky, Kevin t. Smith, Alexey Yakubovich, Wiley, ISBN: 9788126551071, 2015.
14. http://www.johndcook.com/R_language_for_programmers.html
15. http://bigdatauniversity.com/
16. http://home.ubalt.edu/ntsbarsh/stat-data/topics.htm#rintroduction

Third Year B.C.A. (Under Science) Semester VI

BCA – 605 Course **Title: Android Programming Lab**
Course I

Total Contact Hours: 48 hrs. **Total Credits: 04**

Total Marks: 100

Note that these are only sample assignments. Teachers may conduct practicals by preparing similar types of examples

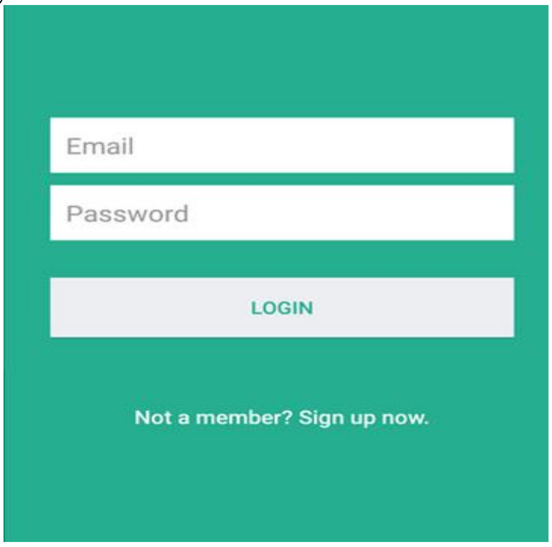
Sample Assignments on Android Programming

Assignment 1: Introduction to Android

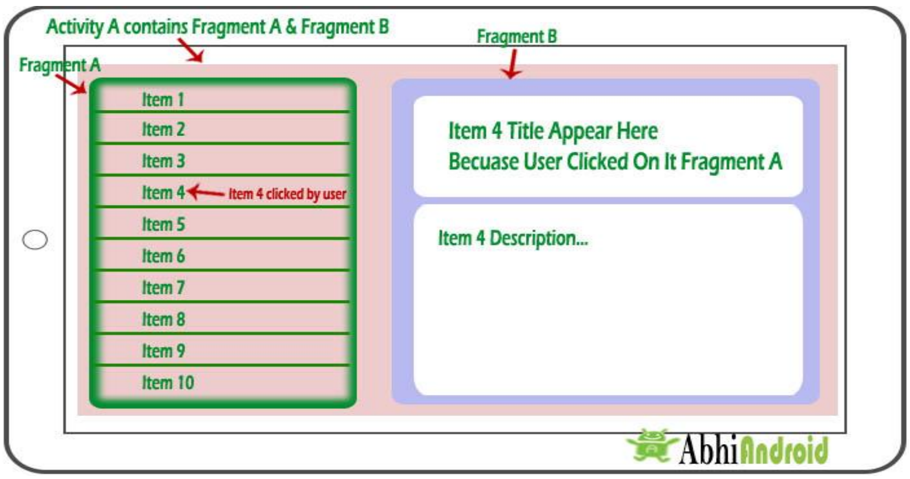
1. Install Android Studio and build simple Hello World application.

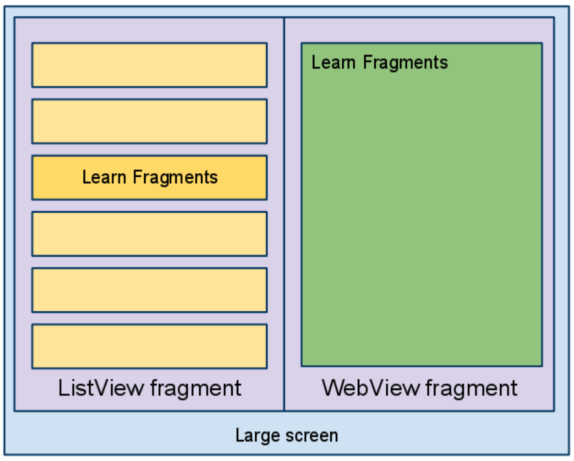
Assignment 2: Activities, Fragments and Intents

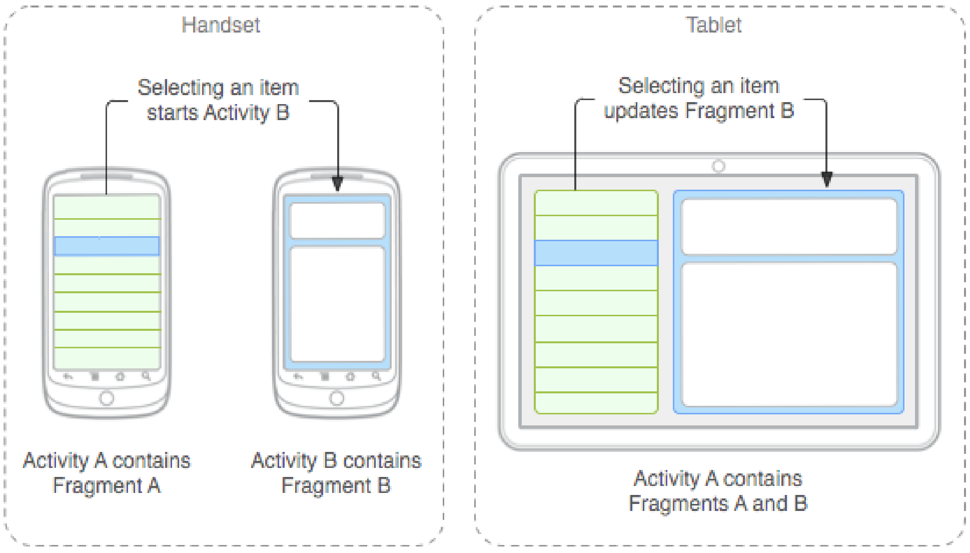
1. Design Login Activity shown below.



2. Create application to display details of selected list item on second activity (Use Fragmentation).



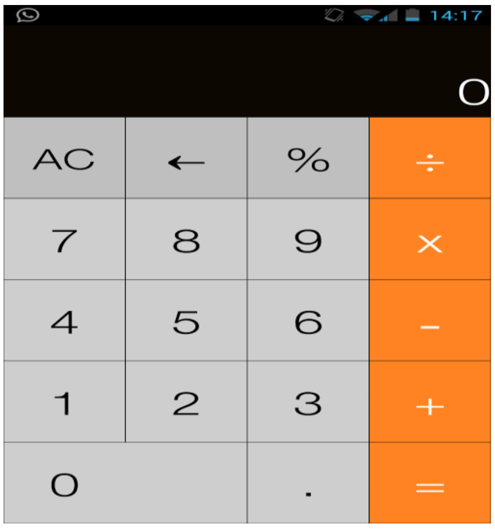



Handset
Selecting an item starts Activity B
Activity A contains Fragment A
Activity B contains Fragment B
Tablet
Selecting an item updates Fragment B
Activity A contains Fragments A and B

3. Create first activity to accept information like first name, last name, date of birth, email-id and display all information on second activity when user click on submit button.

Assignment 3: Android User Interface and Event Handling

1. Create the simple calculator shown below. Also, perform appropriate operations.

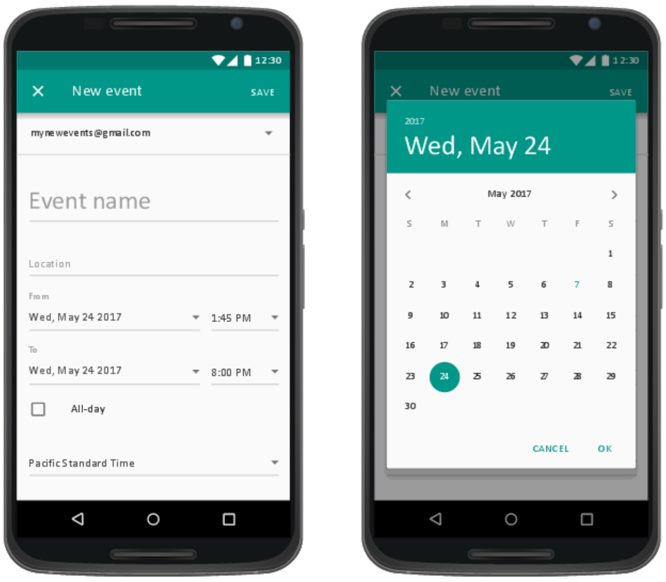


2. Create application to calculate GPA.
3. Create chat application.

Assignment 4: Designing Your User Interface with Views

1. Create a custom "Contact" layout to hold multiple pieces of information, including: Photo, Name, Contact Number, E-mail id.

2. Create application to demonstrate date and time picker.



Assignment 5: Displaying Pictures and Menus

1. Construct an app that toggles a light bulb on and off when the user clicks on toggle button.
2. Create application as shown below.

3. Create gallery application to display all images date wise (Use Grid View).

Assignment 6: Databases – SQLite

1. Create login activity (referAssignment 2 Example 1). If Email and password matches with database display “login successful” message else display error message.
2. Construct a simple notes list that lets the user add new notes but not edit them. Demonstrates the basics of ListActivity.Use a SQLite database to store the notes.
3. Create tables: Course (id, name, instructor) and Student (id, name).
   Course and Student have a many to many relationship. Create a GUI based system for performing the following operations on the tables:
    1. Course: Add Course, View All students of a specific course
    2. Student: Add Student, Delete Student, View All students, Search student

Assignment 7: Messaging and E-mail

1. Create application to send and receive messages.
2. Create application to send email with validation.
3. Create application to send email with attachment.

Assignment 8: Location-Based Services and Google Map

1. Write a program to find the current location of an Android device and display details of the place like Street name, city with Geocoding.
2. Write a program to track android device usingGoogle Maps.
3. Write a program todraw path along a route in Google map.

Third Year B.C.A. (Under Science) Semester VI

Course BCA606 **Title: Python Lab Cource II**

Total Contact Hours: 48 hrs. **Total Credits: 04**

Total Marks: 100

Python Assignments:

1. Create a program that asks the user to enter their name and their age. Print out a message addressed to them that tells them the year that they will turn 100 years old.
2. Write a program to check whether the number is even or odd, print out an appropriate message to the user.
3. Write a program which will find all such numbers which are divisible by 7.
4. Write a program which can compute the factorial of a given numbers.
5. Write a program that prints out all the elements of the list that are less than 10.
6. Write a program that returns a list that contains only the elements that are common between the lists (without duplicates). Make sure your program works on two lists of different sizes.
7. Write a program which accepts a sequence of comma-separated numbers from console and generate a list and a tuple which contains every number.Given the input: 34,67,55,33,12,98

   Then, the output should be:

   ```
   ['34', '67', '55', '33', '12', '98']
   ('34', '67', '55', '33', '12', '98')
   ```

8. Make a two-player Rock-Paper-Scissors game. (*Hint: Ask for player plays (using input), compare them, print out a message of congratulations to the winner, and ask if the players want to start a new game*) Rules:
   Rock beats scissors
   Scissors beats paper
   Paper beats rock
9. To determine whether the number is prime or not.
10. To check whether a number is palindrome or not. (using recursion and without recursion).
11. Write a program (function!) that takes a list and returns a new list that contains all the elements of the first list minus all the duplicates.Write two different functions to do this - one using a loop and constructing a list, and another using sets.
12. Write a program that asks the user how many Fibonnaci numbers to generate and then generates them.
13. Write a program (using functions!) that asks the user for a long string containing multiple words. Print back to the user the same string, except with the words in backwards order. E.g " I am tybca student" is :"student tybca am I"
14. Write a password generator in Python. Be creative with how you generate passwords - strong passwords have a mix of lowercase letters, uppercase letters, numbers, and symbols. The passwords should be random, generating a new password every time the user asks for a new password.
15. Write a program to implement binary search to search the given element using function.
16. Given a `.txt` file that has a list of a bunch of names, count how many of each name there are in the file, and print out the results to the screen.

17. Implement a function that takes as input three variables, and returns the largest of the three.(do not use max function)
18. Create a dictionary (in your file) of names and birthdays. When you run your program it should ask the user to enter a name, and return the birthday of that person back to them.
19. Write a program that takes a list of numbers (for example, `a = [5, 10, 15, 20, 25]`) and makes a new list of only the first and last elements of the given list.
20. Write a program that accepts sequence of lines as input and prints the lines after making all characters in the sentence capitalized.
21. Write a program that accepts a sentence and calculate the number of letters and digits.
22. Write a program that accepts a sentence and calculate the number of upper case letters and lower case letters.

String:
A string is a sequence of characters. The string is a sequence of Unicode character in Python. Unicode was introduced to include every character in all languages and bring uniformity in encoding.
Strings can be created by enclosing characters inside a single quote or double quotes. Even triple quotes can be used in Python but generally used to represent multiline strings and docstrings.

```
# All of the following are equivalent
my_string = 'Hello'
print(my_string)
my_string = "Hello"
print(my_string)
my_string = '''Hello'''
print(my_string)
# triple quotes string can extend multiple lines
my_string = """Hello, welcome to
           the world of Python"""
print(my_string)
```

The output of *stringm.py* will be:

```
Hello
Hello
Hello
Hello, welcome to
           the world of Python
```

To access characters in a string:
We can access individual characters using indexing and a range of characters using slicing. Index starts from 0. Trying to access a character out of index range will raise an IndexError. The index must be an integer. We can't use float or other types, this will result into TypeError. Python allows negative indexing for its sequences.The index of -1 refers to the last item, -2 to the second last item and so on. We can access a range of items in a string by using the slicing operator (colon).

```
str = 'programing'
print('str = ', str)
```

#first character
print('str[0] = ', str[0])
#last character
print('str[-1] = ', str[-1])
#slicing 2nd to 5th character
print('str[1:5] = ', str[1:5])
#slicing 6th to 2nd last character
print('str[5:-2] = ', str[5:-2])
Update string:
The existing string can be update by (re)assigning a variable to another string. The new value can be related to its previous value or to a completely different string altogether. For example −

```
var1 = 'Hello World!'
print "Updated String :- ", var1[:6] + 'Python'
```

output:

```
Updated String :-  Hello Python
```

Python includes the following built-in methods to manipulate strings −

Sr.No.	Methods with Description
1	**capitalize()** Capitalizes first letter of string
2	**center(width, fillchar)** Returns a space-padded string with the original string centered to a total of width columns.
3	**count(str, beg= 0,end=len(string))** Counts how many times str occurs in string or in a substring of string if starting index beg and ending index end are given.
4	**decode(encoding='UTF-8',errors='strict')** Decodes the string using the codec registered for encoding. encoding defaults to the default string encoding.
5	**encode(encoding='UTF-8',errors='strict')** Returns encoded string version of string; on error, default is to raise a ValueError unless errors is given with 'ignore' or 'replace'.
6	**endswith(suffix, beg=0, end=len(string))** Determines if string or a substring of string (if starting index beg and ending index end are given) ends with suffix; returns true if so and false otherwise.
7	**expandtabs(tabsize=8)** Expands tabs in string to multiple spaces; defaults to 8 spaces per tab if tabsize not provided.
8	**find(str, beg=0 end=len(string))** Determine if str occurs in string or in a substring of string if starting index beg

	and ending index end are given returns index if found and -1 otherwise.
9	**index(str, beg=0, end=len(string))** Same as find(), but raises an exception if str not found.
10	**isalnum()** Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise.
11	**isalpha()** Returns true if string has at least 1 character and all characters are alphabetic and false otherwise.
12	**isdigit()** Returns true if string contains only digits and false otherwise.
13	**islower()** Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise.
14	**isnumeric()** Returns true if a unicode string contains only numeric characters and false otherwise.
15	**isspace()** Returns true if string contains only whitespace characters and false otherwise.
16	**istitle()** Returns true if string is properly "titlecased" and false otherwise.
17	**isupper()** Returns true if string has at least one cased character and all cased characters are in uppercase and false otherwise.
18	**join(seq)** Merges (concatenates) the string representations of elements in sequence seq into a string, with separator string.
19	**len(string)** Returns the length of the string
20	**ljust(width[, fillchar])** Returns a space-padded string with the original string left-justified to a total of width columns.
21	**lower()** Converts all uppercase letters in string to lowercase.
22	**lstrip()** Removes all leading whitespace in string.

23	**maketrans()** Returns a translation table to be used in translate function.
24	**max(str)** Returns the max alphabetical character from the string str.
25	**min(str)** Returns the min alphabetical character from the string str.
26	**replace(old, new [, max])** Replaces all occurrences of old in string with new or at most max occurrences if max given.
27	**rfind(str, beg=0,end=len(string))** Same as find(), but search backwards in string.
28	**rindex(str, beg=0, end=len(string))** Same as index(), but search backwards in string.
29	**rjust(width,[, fillchar])** Returns a space-padded string with the original string right-justified to a total of width columns.
30	**rstrip()** Removes all trailing whitespace of string.
31	**split(str="", num=string.count(str))** Splits string according to delimiter str (space if not provided) and returns list of substrings; split into at most num substrings if given.
32	**splitlines(num=string.count('\n'))** Splits string at all (or num) NEWLINEs and returns a list of each line with NEWLINEs removed.
33	**startswith(str, beg=0,end=len(string))** Determines if string or a substring of string (if starting index beg and ending index end are given) starts with substring str; returns true if so and false otherwise.
34	**strip([chars])** Performs both lstrip() and rstrip() on string.
35	**swapcase()** Inverts case for all letters in string.
36	**title()** Returns "titlecased" version of string, that is, all words begin with uppercase and the rest are lowercase.
37	**translate(table, deletechars="")**

	Translates string according to translation table str(256 chars), removing those in the del string.
38	**upper()** Converts lowercase letters in string to uppercase.
39	**zfill (width)** Returns original string leftpadded with zeros to a total of width characters; intended for numbers, zfill() retains any sign given (less one zero).
40	**isdecimal()** Returns true if a unicode string contains only decimal characters and false otherwise.

1. Write a Python function to calculate the factorial of a number (a non-negative integer). The function accepts the number as an argument.
2. Write a Python program to converting an Integer to a string in any base.
3. Write a Python program of recursion list sum.
4. Write a Python program to solve the Fibonacci sequence using recursion.
5. Write a Python program to get the sum of a non-negative integer.
6. Write a Python program to calculate the value of 'a' to the power 'b'
7. Write a Python program to find the greatest common divisor (gcd) of two integers
8. Write a Python function that takes a list and returns a new list with unique elements of the first list.
9. Write a Python function to check whether a number is perfect or not
10. Write a Python program to read a file line by line store it into an array.
11. Write a Python program to count the number of lines in a text file.
12. Write a Python program to count the frequency of words in a file.
13. Write a Python program to copy the contents of a file to another file
14. Write a Python program to read a random line from a file
15. Write a Python class to implement pow(x, n).
16. Write a Python class to reverse a string word by word.
    Input string : 'hello .py'
    Expected Output : '.py hello'
17. Write a Python class named Rectangle constructed by a length and width and a method which will compute the area and perimeter of a rectangle. –
18. Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle

Third Year B.C.A. (Under Science) Semester VI

Course Code: BCA607 **Course Title: Introduction to Green Computing**

Total Contact Hours: 24 hrs. (30ectures) **Total Credits: 02** **Total Marks: 50**

Teaching Scheme: Theory- 05 Lect./ Week

Course Objectives:

1. Building more energy-efficient computing systems as well as building computing technology that increases energy-efficiency of other physical systems.
2. Investigate recent advances in the broad realm of green technologies to save energy and reduce the carbon footprint of modern computing and engineered systems.
3. A holistic coverage is given ranging from single device issues to algorithms for reducing power consumption of data centers, transportation systems, and smart buildings.

Unit No.	Contents	No. of Lecctures
Unit 1	**1. Introduction to Green Computing** Websites &statistics How bad the energy crisis really is?	**04**
Unit 2	**2. Reducing the IT footprint** What really contributes to the footprint (from machine manufacturing to disposal)? Saving energy on a single machine Saving energy in networking and other components Saving energy in clusters and data centers Saving energy on data center cooling	**10**
Unit 3	**3. Computing technology for energy efficiency of other physical systems** Computing technology for greener transportation Computing technology for smarter buildings Carbon footprint calculators: what is my footprint?	**10**
Unit 4	**4. Major green initiatives** Sustainable IT, Green Business, Smarter Plant.	**06**

<u>Reference Books:</u>

1. Green Computing: Tools and Techniques for Saving Energy, Money, and Resources1st Edition by Bud E. Smith(CRC Press)
2. The Green Computing Book by Wu-Chun Feng (CRC Press)
3. Green it for sustainable business practice: An ISEB Foundation Guide by Mark G. O’Neill.