K.S.R. COLLEGE OF ENGINEERING (Autonomous)

Department of Computer Science and Engineering

Subject Name: Python Programming
Subject Code: 18CS043 **Year/Semester:** II/III

Course Outcomes: On completion of this course, the student will be able to

CO1 *Impart the fundamental concepts of python programming.*
CO2 *Know various data structures provided by python library including string, list, dictionary etc.,*
CO3 *To learn to write programs using class.*
CO4 *To study database system for storing and retrieving data.*
CO5 *To learn the concept of Web and GUI design.*

Program Outcomes (POs) and Program Specific Outcomes (PSOs)

A. Program Outcomes (POs)

Engineering Graduates will be able to :

PO1 **Engineering knowledge:** Ability to exhibit the knowledge of mathematics, science, engineering fundamentals and programming skills to solve problems in computer science.

PO2 **Problem analysis:** Identify, formulate, analyze and solve complex engineering problems with the knowledge of computer science. .

PO3 **Design/development of solutions:** Capability to design, implement, and evaluate a computer based system, process, component or program to meet desired needs.

PO4 **Conduct investigations of complex problems:** Potential to conduct investigation of complex problems by methods that include appropriate experiments, analysis and synthesis of information in order to reach valid conclusions.

PO5 **Modern tool Usage:** Ability to create, select, and apply appropriate techniques, resources and modern engineering tools to solve complex engineering problems.

PO6 **The engineer and society:** Skill to acquire the broad education necessary to understand the impact of engineering solutions on a global economic, environmental, social, political, ethical, health and safety.

PO7 **Environmental and sustainability:** Ability to understand the impact of the professional engineering solutions in societal and Environmental contexts and demonstrate the knowledge of, and need for sustainable development.

PO8 **Ethics:** Apply ethical principles and commit to professional ethics and responsibility and norms of the engineering practices.

PO9 **Individual and team work:** Ability to function individually as well as on multi-disciplinary teams.

PO10 **Communication:** Ability to communicate effectively in both verbal and written mode to excel in the career.

PO11 **Project management and finance:** Ability to integrate the knowledge of engineering and management principles to work as a member and leader in a team on diverse projects.

PO12 **Life-long learning:** Ability to recognize the need of technological change by independent and life-long learning.

B. Program Specific Outcomes (PSOs)

PSO1 **Technical Competency:** Develop and Implement computer solutions that accomplish goals to the industry, government or research by exploring new technologies.

PSO2 **Professional Awareness:** Grow intellectually and professionally in the chosen field.

K.S.R. COLLEGE OF ENGINEERING (AUTONOMOUS) - 637 215

Vision of the institution

- We envision to achieve status as an excellent educational institution in the global knowledge hub, making self-learners, experts, ethical and responsible engineers, technologists, scientists, managers, administrators and entrepreneurs who will significantly contribute to research and environment friendly sustainable growth of the nation and the world.

Mission of the institution

- To inculcate in the students self-learning abilities that enable them to become competitive and considerate engineers, technologists, scientists, managers, administrators and entrepreneurs by diligently imparting the best of education, nurturing environmental an social needs.
- To foster and maintain a mutually beneficial partnership with global industries and institutions through knowledge sharing, collaborative research and innovation.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vision of the Department

- To create ever green professionals for software industry, academicians for knowledge cultivation and researchers for contemporary society modernization.

Mission of the Department

- To produce proficient design, code and system engineers for software development.
- To keep updated contemporary technology and fore coming challenges for welfare of the society.

Program Educational Objectives (PEOs)

- Figure out, formulate, analyze typical problems and develop effective solutions by imparting the idea and principles of science, mathematics, engineering fundamentals and computing.
- Competent professionally and successful in their chosen career through life-long learning
- Excel individually or as member of a team in carrying out projects and exhibit social needs and follow professional ethics.

K.S.R. COLLEGE OF ENGINEERING (Autonomous) **R 2018**

SEMESTER – III

		L	T	P	C
18CS043	**PYTHON PROGRAMMING** (Common To CS & EE)	3	0	0	3

Prerequisite: *Basic knowledge of C programming.*

Objectives:

- *To impart the fundamental concepts of python programming.*
- *To know various data structures provided by python library including string, list, dictionary etc.,*
- *To learn to write programs using class.*
- *To study database system for storing and retrieving data.*
- *To learn the concept of Web and GUI design.*

UNIT – I **FUNDAMENTALS OF PYTHON** **[9]**

Introduction to Python – Advantages of Python programming – Variables and Datatypes – Comments – I/O function – Operators – Selection control structures – Looping control structures –Functions: Declaration – Types of arguments – Anonymous functions: Lambda.

UNIT – II **DATA STRUCTURES AND PACKAGES** **[9]**

Strings –List – Tuples – Dictionaries–Sets – Exception Handling: Built-in Exceptions – User-defined exception– Modules and Packages.

UNIT – III **OBJECT ORIENTED PROGRAMMING** **[9]**

Object Oriented Programming basics –Inheritance and Polymorphism – Operator Overloading and Overriding – Get and Set Attribute Values – Name Mangling –Duck Typing – Relationships.

UNIT – IV **FILES AND DATA BASES** **[9]**

File I/O operations – Directory Operations – Reading and Writing in Structured Files: CSV and JSON – Data manipulation using Oracle, MySQL and SQLite.

UNIT – V **GUI AND WEB** **[9]**

UI design: Tkinter – Events – Socket Programming – Sending email – CGI: Introduction to CGI Programming, GET and POST Methods, File Upload.

Total = 45 Periods

Course Outcomes: On Completion of this course, the student will be able to

- *Illustrate basic concepts of python programming.*
- *Apply the necessary data structures includes list, tuple and dictionary in the required fields.*
- *Analyze, design and implement the problems using OOPs technology*
- *Demonstrate the simple file operations*
- *Design web site using GUI.*

Text Books :

1 Mark Lutz, “Learning Python”, O'Reilly Media, 5th Edition, 2013

2 Wesley J.Chun, “Core Python Programming”, Pearson Education, 2nd Edition, 2017

References :

1 Bill Lubanovic, "Introducing Python Modern Computing in Simple Packages", O'Reilly Media, 1st Edition 2014

2 David Beazley, Brian K. Jones, "Python Cookbook", O'Reilly Media, 3rd Edition, 2013

3 Mark Lutz, “Python Pocket Reference”, O'Reilly Media, 5th Edition, 2014

4 www.python.org

5 www.diveintopython3.net

6 To practice: www.codeacademy.com and https://codingbat.com/python.

UNIT I - FUNDAMENTALS OF PYTHON (CO1)
Part A (2 Marks)

1. **What is meant by interpreter? (Remembering)**

   An interpreter is a computer program that executes instructions written in a programming language. It can either execute the source code directly or translate the source code in a first step into a more efficient representation and executes this code.

2. **How will you invoke the python interpreter? (Understanding)**

   The Python interpreter can be invoked by typing the command "python" without any parameter followed by the "return" key at the shell prompt.

3. **State the features of python. (Remembering)**

   - Easy to Learn and Use
   - Interpreted Language
   - Cross-platform Language
   - Free and Open Source
   - Object-Oriented Language
   - Extensible and Integrated
   - Large Standard Library
   - GUI Programming Support

4. **What is meant by interactive mode of the interpreter? (Understanding)**

   Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole.

5. **Write a snippet to display "Hello World" in python interpreter. (Applying)**

   In script mode:

   ```
   >>> print "Hello World"
   ```

   ```
   "Hello
   World"
   'Hello
   World'
   ```

6. **What is a value? What are the different types of values? (Understanding)**

   A value is one of the fundamental things – like a letter or a number – that a program manipulates. Its types are: integer, float, boolean, strings and lists.

7. **Define a variable and write down the rules for naming a variable. (Understanding)**

   A name that refers to a value is a variable. Variable names can be arbitrarily long. They can contain both letters and numbers, but they have to begin with a letter. It is legal to use uppercase letters, but it is good to begin variable names with a lowercase letter.

8. **Define keyword and enumerate some of the keywords in Python. (Applying)**

   A keyword is a reserved word that is used by the compiler to parse a program. Keywords cannot be used as variable names. Some of the keywords used in python are: and, del, from, not, while, is, continue.

9. **What are the datatypes supported by python? (Remembering)**

   Python has five standard data types

   - Numbers
   - String
   - List
   - Tuple
   - Dictionary

10. **Define statement and what are its types? (Remembering)**

    A statement is an instruction that the Python interpreter can execute. There are two types of statements: print and assignment statement.

11. **What do you meant by an assignment statement? (Understanding)**

    An assignment statement creates new variables and gives them values:
    Eg 1: Message = 'And now for something completely different'
    Eg 2: n = 17

12. **What are I/O statements in Python? (Understanding)**

    The function **input ( )** takes the input from the user and then evaluates the expression, which means Python automatically identifies whether user entered a string or a number or list.
    The **print()** function is used display the output on the screen.
    Example

```
val = input("Enter your value: ")
print(val)
```

13. **How to write comment line in python? (Understanding)**

    Comments are lines that exist in computer programs that are ignored by compilers and interpreters. Including comments in programs makes code more readable for humans as it provides some information or explanation about what each part of a program is doing.
    Comments in Python begin with a hash mark (#) and whitespace character and continue to the end of the line.

```
# This is a comment
```

14. **What is an expression? (Remembering)**

An expression is a combination of values, variables, and operators. An expression is evaluated using assignment operator.

Examples: Y=x + 17

15. **What do you mean by an operand and an operator? Illustrate your answer with relevant example. (Analyzing)**

    An operator is a symbol that specifies an operation to be performed on the operands. The data items that an operator acts upon are called operands. The operators +, -, *, / and ** perform addition, subtraction, multiplication, division and exponentiation. Example: 20+32
    In this example, 20 and 32 are operands and + is an operator.

16. **What are the operators supported by python? (Remembering)**

    Python language supports the following types of operators.

    - Arithmetic Operators
    - Comparison (Relational) Operators
    - Assignment Operators
    - Logical Operators
    - Bitwise Operators
    - Membership Operators
    - Identity Operators

17. **What is the order in which operations are evaluated? Give the order of precedence. (Analyzing)**

    The set of rules that govern the order in which expressions involving multiple operators and operands are evaluated is known as rule of precedence. Parentheses have the highest precedence followed by exponentiation. Multiplication and division have the next highest precedence followed by addition and subtraction.

18. Illustrate the use of * and + operators in string with example. (Understanding)

The * operator performs repetition on strings and the + operator performs concatenation on strings.

Example:

>>> 'Hello*3'

Output: HelloHelloHello

>>>'Hello+World'

Output: HelloWorld

19. Give the reserved words in Python. (Remembering)

Keywords are the reserved words in Python. We cannot use a keyword as variable name, function name or any other identifier. They are used to define the syntax and structure of the Python language.

Here's a list of all keywords in Python Programming

True	False	None	and	as
asset	def	class	continue	break
else	finally	elif	del	except
global	for	if	from	import
raise	try	or	return	pass
nonlocal	in	not	is	lambda

20. Identify the parts of a function in the given example. (Understanding)

```
betty = type("32")
print betty
```

The name of the function is *type*, and it displays the type of a value or variable. The value or variable, which is called the argument of the function, is enclosed in parentheses. The argument is *32*. The function returns the result called return value. The return value is stored in *betty*.

21. What is a local variable? (Remembering)

A variable defined inside a function. A local variable can only be used inside its function.

22. What is the output of the following? (Evaluating)

```
float(32)
float("3.14159")
```

Output:

a. 32.0 The float function converts integers to floating-point numbers.

b. 3.14159 The float function converts strings to floating-point numbers.

23. What do you mean by flow of execution? (Understanding)

In order to ensure that a function is defined before its first use, we have to know the order in which statements are executed, which is called the flow of execution. Execution always begins at the first statement of the program. Statements are executed one at a time, in order from top to bottom.

24. Write down the output for the following program. ? (Evaluating)

```
first = 'throat'
second  'warbler'
print first + second
```

Output:

throatwarbler

25. Explain the concept of floor division. (Understanding)

The operation that divides two numbers and chops off the fraction part is known as floor division.

26. What is type coercion? Give example. (Applying)

Automatic method to convert between data types is called type coercion. For mathematical operators, if any one operand is a float, the other is automatically converted to float.

Eg:

```
minute = 59
minute / 60.00.983333333333
```

27. Write a math function to perform √2 / 2. (Applying)

```
math.sqrt(2) / 2.0
        0.707106781187
```

28. Define Boolean expression with example. (Understanding)

A boolean expression is an expression that is either true or false. The values true and false are called Boolean values.

Eg :

```
5 == 5
False
```

True and False are special values that belongs to the type bool; they are not strings:

29. What are the different types of operators? (Remembering)

- Arithmetic Operator (+, -, *, /, %, **, //)
- Relational operator (== , !=, <>, < , > , <=, >=)
- Assignment Operator (=, += , *= , - =, /=, %= ,**=)
- Logical Operator (AND, OR, NOT)
- Membership Operator (in, not in)
- Bitwise Operator (& (and), | (or) , ^ (binary Xor), ~(binary 1's complement , << (binary left shift), >> (binary right shift))
- Identity(is, is not)

30. Explain modulus operator with example. (Applying)

The modulus operator works on integers and yields the remainder when the first operand is divided by the second. In Python, the modulus operator is a percent sign (%). The syntax is the same as for other operators:

Eg:

```
remainder = 7 % 3
print remainder
1
```

So 7 divided by 3 is 2 with 1 left over.

31. Explain relational operators. (Understanding)

The == operator is one of the relational operators; the others are:

X! = y # x is not equal to y

x > y # x is greater than y

x < y # x is less than y

x >= y # x is greater than or equal to y

x <= y # x is less than or equal to y

32. Explain Logical operators (Applying)

There are three logical operators: and, or, and not. For example, x > 0 and x < 10 is true only if x is greater than 0 and less than 10. n%2 == 0 or n%3 == 0 is true if either of the conditions is true, that is, if the number is divisible by 2 or 3. Finally, the not operator negates a Boolean expression, so not(x > y) is true if x > y is false, that is, if x is less than or equal to y. Non-zero number is said to be true in Boolean expressions.

33. What is conditional execution? (Remembering)

The ability to check the condition and change the behavior of the program accordingly is called conditional execution. Example:

If statement:

The simplest form of if statement is:

Syntax:

if(cond):
statement:

Eg:

```
if x > 0:
print 'x is positive'
```

The boolean expression after 'if' is called the condition. If it is true, then the indented statement gets executed. If not, nothing happens.

34. What is alternative execution? (Understanding)

A second form of if statement is alternative execution, that is, if …else, where there are two possibilities and the condition determines which one to execute.

Eg:

```
if x%2 == 0:
print 'x is even'
else:
print 'x is odd'
```

35. What are chained conditionals? (Understanding)

Sometimes there are more than two possibilities and we need more than two branches. One way to express a computation like that is a chained conditional:

Eg:

```
if x < y:
print 'x is less than y'
elif x > y:
print 'x is greater than y'
else:
print 'x and y are equal'
```

elif is an abbreviation of "else if." Again, exactly one branch will be executed. There is no limit on the number of elif statements. If there is an else clause, it has to be at the end, but there doesn't have to be one.

36. Explain while loop with example. (Understanding)

Eg:

```
def countdown(n):
while n > 0:
print n
n = n-1
print 'Blastoff!'
```

More formally, here is the flow of execution for a while statement:

Evaluate the condition, yielding True or False.

If the condition is false, exit the while statement and continue execution at the next statement.

If the condition is true, execute the body and then go back to step 1

37. Explain 'for loop' with example. (Understanding)

The general form of a for statement is

Syntax:

for variable in sequence:

code block

Eg:

```
x = 4
for i in range(0, x):
print i
```

38. What is a break statement? (Remembering)

When a break statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.

Eg:

```
while True:
line = raw_input('>')
if line == 'done':
break
print line
print'Done!'
```

39. What is a continue statement? (Remembering)

The continue statement works somewhat like a break statement. Instead of forcing termination, it forces the next iteration of the loop to take place, skipping any code in between.

Eg:

```
for num in range(2,10):
if num%2==0;
print “Found an even number”, num
continue
print “Found a number”, num
```

40. Differentiate global and local scope. (Evaluating)

The scope of a variable refers to the places that we can see or access a variable. If we define a variable on the top of the script or module, the variable is called global variable. The variables that are defined inside a class or function is called local variable.

Eg:

```
def my_local():
a=10
print(“This is local variable”)
```

Eg:

```
a=10
def my_global():
print(“This is global variable”)
```

41. What is the purpose of pass statement? (Remembering)

Using a pass statement is an explicit way of telling the interpreter to do nothing.

Eg:

```
def bar():
pass
```

If the function bar() is called, it does absolutely nothing.

42. What is function call? (Analyzing)

A function is a named sequence of statements that performs a computation. When we define a function, we specify the name and the sequence of statements. Later, we can "call" the function by its name called as function call.

43. Give the syntax of function definition. (Understanding)

```
def NAME( LIST OF PARAMETERS ):
.
.

STATEMENTS
```

44. What are function arguments in python? (Understanding)

- Default arguments
- Required arguments
- Keyword arguments
- Variable number of arguments

45. What is the use of return statement? (Understanding)

Return gives back or replies to the caller of the function. The return statement causes our function to exit and hand over back a value to its caller.

Eg:

```
def area(radius):
temp = math.pi * radius**2
return temp
```

46. What is recursion? (Understanding)

The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called as recursive function.

Eg:

```
def factorial(n):
if n == 1:
return 1
else:
return n * factorial(n-1)
```

47. Define Anonymous function? (Understanding)

Anonymous function is a function that is defined without a name. While normal functions are defined using the def keyword, in Python anonymous functions are defined using the lambda keyword. Hence, anonymous functionsare also called lambda functions.

48. **Write the syntax of lambda function. (Applying)**

Syntax

```
lambda arguments : expression
```

Lambda functions can have any number of arguments but only one expression. The expression is evaluated and returned. Lambda functions can be used wherever function objects are required.

```
double = lambda x: x * 2

# Output: 10
print(double(5))
```

Part B(16 Marks)

1. Explain built-in data types of python**.(Remembering)**
2. List down the different types of operators with suitable examples. **(Understanding)**
3. Explain the operator precedence and associativity of operators. **(Analyzing)**
4. Explain conditional alternative and chained conditional. **(Applying)**
5. Briefly discuss in detail about function prototyping in python. With suitable example program. **(Applying)**
6. Describe the syntax and rules involved in the return statement in python. **(Understanding)**
7. Describe in detail about lambda functions or anonymous function. **(Understanding)**
8. Compare different types of function arguments. **(Evaluating)**
9. What type of parameter passing is used in Python? Explain with sample programs. **(Understanding)**
10. Write a Python program for a) selection sort b) insertion sort. **(Creating)**

UNIT II - DATA STRUCTURES AND PACKAGES (CO2)

Part A (2 Marks)

1. **Compare string and string slices. (Evaluating)**

A string is a sequence of character.

Eg: fruit = ‘banana’

String Slices :

A segment of a string is called string slice, selecting a slice is similar to selecting a character.

Eg: >>> s ='Monty Python'
print s[0:5]
Monty
print s[6:12]
Python

2. **Mention a few string functions. (Analyzing)**

s.captilize() – Capitalizes first character of string
s.count(sub) – Count number of occurrences of sub in
string

s.lower() – converts a string to lower case
s.split() – returns a list of words in string

3. **What is a list? (Remembering)**

A list is an ordered set of values, where each value is identified by an index. The values that make up a list are called its elements. Lists are similar to strings, which are ordered sets of characters, except that the elements of a list can have any type.

4. **Solve a)[0] * 4 and b) [1, 2, 3] * 3.**

>>> [0] * 4 [0, 0, 0, 0]

>>> [1, 2, 3] * 3 [1, 2, 3, 1, 2, 3, 1, 2, 3]

5. **Let list = [’a’, ’b’, ’c’, ’d’, ’e’, ’f’]. Find a) list[1:3] b) t[:4] c) t[3:] . (Evaluating)**

>>> list = [’a’, ’b’, ’c’, ’d’, ’e’, ’f’]

>>> list[1:3] [’b’, ’c’]

>>> list[:4] [’a’, ’b’, ’c’, ’d’]

>>> list[3:] [’d’, ’e’, ’f’]

6. **Mention any 5 list methods. (Understanding)**

append() ,extend () ,sort(), pop(),index(),insert and remove()

7. **What is List mutability in Python? Give an example. (Understanding)**

Python represents all its data as objects. Some of these objects like **lists** and dictionaries are **mutable**, i.e., their content can be changed without changing their identity. Other objects like integers, floats, strings and tuples are objects that cannot be changed. Example:

```
>>> numbers = [17, 123]

>>> numbers[1] = 5

>>> print numbers [17, 5]
```

8. **What is aliasing in list? Give an example. (Understanding)**

An object with more than one reference has more than one name, then the object is said to be aliased. Example: If *a* refers to an object and we assign b = a, then both variables refer to the same object:

```
>>> a = [1, 2, 3]

>>> b = a

>>> b is a True
```

9. **Define cloning in list. (Remembering)**

In order to modify a list and also keep a copy of the original, it is required to make a copy of the list itself, not just the reference. This process is called cloning, to avoid the ambiguity of the word “copy”.

10. **Explain List parameters with an example. (Understanding)**

Passing a list as an argument actually passes a reference to the list, not a copy of the list. For example, the function head takes a list as an argument and returns the first element:

```
def head(list):
return list[0]
output:
        >>> numbers = [1, 2, 3]

        >>> head(numbers)
```

11. **Write a program in Python to delete first element from a list. (Applying)**

```
def deleteHead(list): del list[0]
```

Here’s how deleteHead is used:

```
>>> numbers = [1, 2, 3]
>>> deleteHead(numbers)
>>> print numbers [2, 3]
```

12. **What is tuple? (Remembering)**

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

parentheses, whereas lists use square brackets. Creating a tuple is as simple as putting different comma-separated values. Comma-separated values between parentheses can also be used.

Example: tup1 = ('physics', 'chemistry', 1997, 2000);

13. How to return tuples as values? (Understanding)

A function can only return one value, but if the value is a tuple, the effect is the same as returning multiple values. For example, if we want to divide two integers and compute the quotient and remainder, it is inefficient to compute x/y and then x%y. It is better to compute them both at the same time.

```
>>>  t = divmod(7, 3)
 >>>  print t (2, 1)
```

14. What is the benefit of using tuple assignment in Python? (Applying)

It is often useful to swap the values of two variables. With conventional assignments a temporary variable would be used. For example, to swap a and b:

```
temp = a
a = b
b = temp
 a, b = b, a
```

15. Define key-value pairs. (Understanding)

The elements of a dictionary appear in a comma-separated list. Each entry contains an index and a value separated by a colon. In a dictionary, the indices are called keys, so the elements are called key-value pairs.

16. Define dictionary with an example. (Understanding)

A dictionary is an associative array (also known as hashes). Any key of the dictionary is associated (or mapped) to a value. The values of a dictionary can be any Python data type. So dictionaries are unordered key-value-pairs.

Example:

```
>>> eng2sp = {}    # empty dictionary

 >>>  eng2sp['one'] = 'uno'
>>>  eng2sp['two'] = 'dos'
```

17. List two dictionary operations. (Understanding)

Del -removes key-value pairs from a dictionary

Len - returns the number of key-value pairs

18. Define dictionary methods with an example. (Understanding)

A method is similar to a function—it takes arguments and returns a value— but the syntax is different. For example, the keys method takes a dictionary and returns a list of the keys that appear, but instead of the function syntax keys(eng2sp), method syntax eng2sp.keys() is used.

```
>>>  eng2sp.keys() ['one', 'three', 'two']
```

19. Define List Comprehension. (Understanding)

List comprehensions apply an **arbitrary expression** to items in an iterable rather than applying function. It provides a compact way of mapping a list into another list by applying a function to each of the elements of the list.

20. Define Set. (Remembering)

A set is a collection which is unordered and unindexed. In Python sets are written with curly brackets.

```
thisset={"apple", "banana", "cherry"}
print(thisset)
```

21. What are the built-in methods of set? (Understanding)

- add() - Adds an element to the set
- clear() - Removes all the elements from the set
- copy() - Returns a copy of the set
- pop() - Removes an element from the set
- remove() - Removes an element from the set

22. Write a Python program to swap two variables. (Creating)

```
x = 5
y = 10
temp = x
x = y
y = temp
print('The value of x after swapping: {}'.format(x))
print('The value of y after swapping: {}'.format(y))
```

23. What are exceptions? (Remembering)

An exception is an error that happens during execution of a program. When that error occurs, Python generate an exception that can be handled, which avoids your program to crash.

24. List the common exception error in python. (Understanding)

- ZeroDivisionError
- ValueError
- NameError
- KeyError
- FloatingPointError

25. How exceptions are handled in python? (Analyzing)

To use exception handling in Python, you first need to have a catch-all except clause. The words "try" and "except" are Python keywords and are used to catch exceptions.

try-except [exception-name] (see above for examples) **blocks**

The code within the try clause will be executed statement by statement.If an exception occurs, the rest of the try block will be skipped and the except clause will be executed.

Syntax

```
try:
    some statements here
except:
    exception handling
```

Example

```
try:
   print 1/0
except ZeroDivisionError:
   print "You can't divide by zero, you're silly."
```

26. **How do you handle the exception inside a program when you try to open a non-existent file? (Analyzing)**

```
filename = raw_input('Enter a file name: ')
try:
f = open (filename, "r")
except IOError:
print 'There is no file named', filename
```

27. **Write the syntax of Try ... except ... else clause. (Understanding)**

The else clause in a try , except statement must follow all except clauses, and is useful for code that must be executed if the try clause does not raise an exception.

```
try:
   data = something_that_can_go_wrong

except IOError:
   handle_the_exception_error

else:
   doing_different_exception_handling
```

28. **What are modules? (Remembering)**

A module is simply a file that defines one or more related functions grouped together. To reuse the functions of a given module, we need to import the module. Syntax: import <modulename>

29. **List the important built-in modules of python. (Remembering)**

- calendar
- cgi
- io
- keyword
- math
- re
- random

30. **Write syntax of import statement. (Understanding)**

The import Statement

Module contents are made available to the caller with the import statement. The import statement takes many different forms, shown below.

import <module_name>

The simplest form is the one already shown above:

```
import <module_name>
```

31. **What is a package? (Remembering)**

Packages are namespaces that contain multiple packages and modules themselves. They are simply directories.

Syntax: from <mypackage> import <modulename>

32. What is the use of dir function? (Analyzing)

Python dir() function attempts to return a list of valid attributes for the given object. If no argument provided, then it returns the list of names in the current local scope. If the module name is passed as an argument this function returns the functions implemented in each module.

33. What is the special file that each package in Python must contain? (Understanding)

Each package in Python must contain a special file called __init__.py

34. State the difference between lists and dictionary. ? (Evaluating)

List is a mutable type meaning that it can be modified whereas dictionary is immutable and is a key value store. Dictionary is not ordered and it requires that the keys are hash table whereas list can store a sequence of objects in a certain order.

35.Write a program in Python returns a list that contains all but the first element of the given list. ? (Evaluating)

```
define tail(list): return list[1:]
```

Here's how tail is used:

```
>>> numbers = [1, 2, 3]
>>> rest = tail(numbers)
 >>>  print rest [2, 3]
```

Part B (16 Marks)

1. Explain string slices and string immutability. **(Understanding)**
2. Explain string functions and methods. **(Understanding)**
3. Explain in detail about lists, list operations and list slices. **(Remembering)**
4. Explain in detail about list methods and list loops with examples. **(Analyzing)**
5. Compare list, tuples and sets. **(Evaluating)**
6. Explain in detail about mutability and tuples with a Python program **(Understanding)**
7. Explain in detail about dictionaries and its operations. **(Understanding)**
8. Describe the usage of python sets and its methods. **(Understanding)**
9. Explain the use of try and catch block in python with its syntax. Also list the standard exceptions in python. **(Understanding)**
10. Briefly describe about modules and packages. **(Remembering)**
11. How to create a module and use it in a python program explain with an example.**(Analyzing)**
12. List out the types of modules, and explain any of the two in detail. **(Remembering)**

UNIT III - OBJECT ORIENTED PROGRAMMING (CO3)

Part A (2 Marks)

1. List OOPs Concepts of Python. (Remembering)

Python is an object-oriented programming language. It allows us to develop applications using an Object Oriented approach. Major principles of object-oriented programming system are given below.

- Object
- Class
- Method
- Inheritance
- Polymorphism
- Data Abstraction
- Encapsulation

2. Define class. (Remembering)

A Class is like an object constructor, or a "blueprint" for creating objects.
Create a Class
To create a class, use the keyword class:
Example
Create a class named MyClass, with a property named x:

```
class MyClass:
  x = 5
```

3. How to create object in python? (Understanding)

The procedure to create an object is similar to a function call.

```
>>> ob = MyClass()
```

This will create a new instance object named ob. We can access attributes of objects using the object name prefix.

4. Define Constructor. (Remembering)

A constructor is a special type of method (function) which is used to initialize the instance members of the class.
Constructors can be of two types.

1. Parameterized Constructor
2. Non-parameterized Constructor

Constructor definition is executed when we create the object of this class. Constructors also verify that there are enough resources for the object to perform any start-up task.

5. How constructors are created in Python? (Understanding)

In python, the method **__init__** simulates the constructor of the class. This method is called when the class is instantiated. We can pass any number of arguments at the time of creating the class object, depending upon **__init__** definition. It is mostly used to initialize the class attributes. Every class must have a constructor, even if it simply relies on the default constructor.

Example

```
class Employee:
  def __init__(self,name,id):
    self.id = id;
    self.name = name;
```

```
    def display (self):
        print("ID: %d \nName: %s"%(self.id,self.name))
    emp1 = Employee("John",101)
    emp2 = Employee("David",102)
     #accessing display() method to print employee 1 information
     emp1.display();
     #accessing display() method to print employee 2 information
     emp2.display();
```

6. **What is operator overloading? (Remembering)**

   Operator Overloading means giving extended meaning beyond their predefined operational meaning. For example operator + is used to add two integers as well as join two strings and merge two lists. It is achievable because '+' operator is overloaded by int class and str class.

7. **How to overload the operators in Python? (Understanding)**

   To perform operator overloading, Python provides some special function or magic function that is automatically invoked when it is associated with that particular operator.

   Example:

```
Class A:
   def __init__(self, a):
      self.a = a

   # adding two objects
   def __add__(self, o):
      return self.a + o.a
ob1 = A(1)
ob2 = A(2)
ob3 = A("Geeks")
ob4 = A("For")

print(ob1 + ob2)
print(ob3 + ob4)
```

8. **Define Inheritance. (Remembering)**

   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 re-usability of the code.

9. **What is the syntax of Inheritance? (Understanding)**

   In python, a derived class can inherit base class by just mentioning the base in the bracket after the derived class name.

   class derived-class(base class):

   <class-suite>

   A class can inherit multiple classes by mentioning all of them inside the bracket. Consider the following syntax.

   class derive-class(<base class 1>, <base class 2>, ... <base class n>):

   <class - suite>

10. What are the types of inheritance in python? (Remembering)

In Python, there are two types of Inheritance:

- Single Inheritance
- Multiple Inheritance
- Multilevel Inheritance

11. Compare the use of isinstance() and issubclass() built-in function? (Evaluating)

The isinstance() method is used to check the relationship between the objects and classes. It returns true if the first parameter, i.e., obj is the instance of the second parameter, i.e., class.

The issubclass(sub, sup) method is used to check the relationships between the specified classes. It returns true if the first class is the subclass of the second class, and false otherwise.

12. Define method overriding in python. (Remembering)

In Python method overriding occurs by simply defining in the child class a method with the same name of a method in the parent class.

```
class Parent(object):

     def __init__(self):

         self.value = 4

     def get_value(self):

         return self.value

 class Child(Parent):

     def get_value(self):

         return self.value + 1
```

Now Child objects behave differently

```
>>> c = Child()

>>> c.get_value()

5
```

13. What are Namespaces and Scope in Python? (Understanding)

A namespace is a system to have a unique name for each and every object in Python. An object might be a variable or a method. Python itself maintains a namespace in the form of a Python dictionary.

Scope refers to the coding region from which particular Python object is accessible. Hence one cannot access any particular object from anywhere from the code, the accessing has to be allowed by the scope of the object.

14. What is the use of Get and Set Attributes in Python? (Understanding)

Attributes of a class are function objects that define corresponding methods of its instances. They are used to implement access controls of the classes.

getattr() – This function is used to access the attribute of object.
hasattr() – This function is used to check if an attribute exist or not.
setattr() – This function is used to set an attribute. If the attribute does not exist, then it would be created.
delattr() – This function is used to delete an attribute. If you are accessing the attribute after deleting it raises error “class has no attribute”

15. **Define Name Mangling. (Remembering)**

A double underscore prefix causes the Python interpreter to rewrite the attribute name in order to avoid naming conflicts in subclasses. This is also called name mangling—the interpreter changes the name of the variable in a way that makes it harder to create collisions when the class is extended later.

16. **What are Method Types in Python? (Understanding)**

1. Decorators
2. Instance methods
3. Class methods
4. Static methods

17. **What is Duck typing in Python? (Understanding)**

Python is strongly but dynamically typed. This means names in code are bound to strongly typed objects at runtime. The only condition on the type of object a name refers to is that it supports the operations required for the particular object instances in the program. For example, I might have two types Person and Car that both support operation "run", but Car also supports "refuel". So long as my program only calls "run" on objects, it doesn't matter if they are Person or Car. This is called "duck typing" after the expression "if it walks like a duck and talks like a duck, it's a duck."

Part B (16 Marks)

1. Write short notes on isinstance() and __init__() **(Understanding)**
2. List the features and explain about different Object Oriented features supported by Python. **(Understanding)**
3. How to declare a constructor method in python? Explain. **(Analyzing)**
4. Explain the concept of method overriding and method overloading with an example. **(Analyzing)**
5. Explain the types of inheritance supported by python. **(Understanding)**
6. How operator overloading is implemented in python? Give an example. **(Understanding)**

UNIT IV - FILES AND DATA BASES (CO4)

Part A (2 Marks)

1. **What is a text file? (Remembering)**

   A text file is a file that contains printable characters and whitespace, organized in to lines separated by newline characters.

2. **Write the syntax of file open function in python. (Understanding)**

   Before read or write a file, it is necessary to open it using Python's built-in *open()* function. This function creates a **file** object, which would be utilized to call other support methods associated with it.

   Syntax

   file object = open(file_name [, access_mode][, buffering])

   Here are parameter details −

   - **file_name** − The file_name argument is a string value that contains the name of the file that you want to access.
   - **access_mode** − The access_mode determines the mode in which the file has to be opened, i.e., read, write, append, etc. A complete list of possible values is given below in the table. This is optional parameter and the default file access mode is read (r).
   - **buffering** − If the buffering value is set to 0, no buffering takes place. If the buffering value is 1, line buffering is performed while accessing a file. If you specify the buffering value as an integer greater than 1, then buffering action is performed with the indicated buffer size. If negative, the buffer size is the system default(default behavior).

3. **List the different modes of opening a file. (Remembering)**

   - **r -** Opens a file for reading only.
   - **rb -** Opens a file for reading only in binary format.
   - **r+ -** Opens a file for both reading and writing.
   - **rb+ -** Opens a file for both reading and writing in binary format.
   - **w -** Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
   - **wb -**Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
   - **w+ -** Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
   - **wb+ -** Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
   - **a -** Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
   - **Ab -** Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
   - **a+ -** Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.
   - **ab+ -** Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.

4. **Write the syntax of closing a file. (Understanding)**

   The close() method of a *file* object flushes any unwritten information and closes the file object, after which no more writing can be done.

Python automatically closes a file when the reference object of a file is reassigned to another file. It is a good practice to use the close() method to close a file.

Syntax

```
fileObject.close();
```

5. **Write a python program that writes "Hello world" into a file. (Creating)**

```
file.f=open("ex88.txt",'w')
f.write("hello world")
f.close()
```

6. **Write a python program that counts the number of words in a file. (Creating)**

```
file.f=open("test.txt","r")
content =f.readline(20)
words =content.split()
print(words)
```

7. **What are the two arguments taken by the open() function? (Understanding)**

The open function takes two arguments : name of the file and the mode of operation.

Example: f = open("test.dat","w")

8. **What is a file object? (Understanding)**

A file object allows us to use, access and manipulate all the user accessible files. It maintains the state about the file it has opened.

9. **What information is displayed if we print a file object in the given program? (Understanding)**

```
f= open("test.txt","w")
print f
```

The name of the file, mode and the location of the object will be displayed.

10. **How do you delete a file in Python? (Understanding)**

The remove() method is used to delete the files by supplying the name of the file to be deleted as argument.

Syntax: os.remove(filename)

11. **What is Directory in Python? (Remembering)**

A directory or folder is a collection of files and sub directories. Python has the **os** module, which provides us with many useful methods to work with directories

12. **List the directory operations in python. (Remembering)**

- Getting the Current Working Directory - getcwd()
- List Directory Contents - listdir()
- Create a new Directory/Folder - mkdir()
- Creating Subdirectories – makedirs()
- Deleting an empty Directory/Folder - rmdir()
- Renaming a directory/folder - os.rename()

13. **What is CSV format? (Understanding)**

CSV (Comma Separated Values) is a most common file format that is widely supported by many platforms and applications. It is easier to export data as a csv dump from one system to another system. In Python it is simple to read data from csv file and export data to csv. The csv package comes with very handy methods and arguments to read and write csv file.

Example:

```
User,Country,Age
Alex,US,25
Ben,US,24
Dennis,UK,25
Yuvi,IN,24
```

14. **What is JSON format? (Understanding)**

JSON(Java Script Object Notation) is a lightweight text based data-interchange format which is completely language independent. It is based on JavaScript. Easy to understand, manipulate and generate. In Python there are lot of packages to simplify working with json. We are going to use json module in this tutorial.

Example:

```
{
   "Name" : "Alex",
   "City" : "Chennai",
   "Country" : "India",
   "Age" : 25,
   "Skills" : [ "Java", "JSP"]
}
```

15. **Write the steps to convert CSV into JSON format. (Applying)**
    - Get paths to both input csv file, output json file and json formatting via Command line arguments
    - Read CSV file using Python CSV DictReader
    - Convert the csv data into JSON or Pretty print JSON if required
    - Write the JSON to output file

16. **Define SQLite. (Remembering)**

SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine.

17. **What are procedure to query data in SQLite from Python? (Understanding)**

    - First, establish a connection to the SQLite database by creating a Connection object.
    - Next, create a `Cursor` object using the cursor method of the `Connection` object.
    - Then, execute the SELECT statement.
    - After that, call the `fetchall()` method of the cursor object to fetch the data.
    - Finally, loop the cursor and process each row individually.

Part B (16 Marks)

1. Write notes on working of file open() function in python.(Understanding)
2. Describe syntax of opening and closing a file with example. (Understanding)
3. Tabulate the different methods of opening a file and explain. (Analyzing)
4. Write a python program to merge the contents of two files into a single file. (Creating)
5. Explain the file I/O operations of python. (Remembering)
6. Read a text file in Python and print no. of lines and no. of unique words. (Creating)
7. Explain various directory operations of python. (Understanding)
8. Write a python code to create an employee database and perform insert, update, select and delete operation. (Applying)

UNIT V - GUI AND WEB (CO5)
Part A (2 Marks)

1. **What is the use of Tkinter? (Understanding)**
   Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.
2. **List the procedure for creating a GUI application using Tkinter. (Understanding)**

   a. Import the *Tkinter* module.
   b. Create the GUI application main window.
   c. Add one or more of the above-mentioned widgets to the GUI application.
   d. Enter the main event loop to take action against each event triggered by the user.

3. **What are the main methods available for creating GUI application using of Tkinter? (Understanding)**

   a. To create a main window, tkinter offers a method 'Tk(screenName=None, baseName=None, className='Tk', useTk=1)'.
   m=tkinter.Tk() where m is the name of the main window object
   b. There is a method known by the name mainloop() is used when you are ready for the application to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event till the window is not closed.
   m.mainloop()

4. **List the major widgets of Tkinter? (Understanding)**

Widget	Description
Button	Similar to a Label but provides additional functionality for mouse-overs, presses, and releases, as well as keyboard activity/events
Canvas	Provides ability to draw shapes (lines, ovals, polygons, rectangles); can contain images or bitmaps
Checkbutton	Set of boxes, of which any number can be "checked"
Entry	Single-line text field with which to collect keyboard input
Frame	Pure container for other widgets
Label	Used to contain text or images
LabelFrame	Combo of a label and a frame but with extra label attributes
Listbox	Presents the user with a list of choices from which to choose
Menu	Actual list of choices "hanging" from a Menubutton from which the user can choose
Menubutton	Provides infrastructure to contain menus (pulldown, cascading, etc.)
Message	Similar to a Label, but displays multiline text
PanedWindow	A container widget with which you can control other widgets placed within it
Radiobutton	Set of buttons, of which only one can be "pressed"
Scale	Linear "slider" widget providing an exact value at current setting; with defined starting and ending values
Scrollbar	Provides scrolling functionality to supporting widgets, for example, Text, Canvas, Listbox, and Entry
Spinbox	Combination of an entry with a button letting you adjust its value
Text	Multiline text field with which to collect (or display) text from user
Toplevel	Similar to a Frame, but provides a separate window container

5. **What is the standard attributes of widgets? (Remembering)**

   Dimensions
   Colors
   Fonts
   Anchors
   Relief styles
   Bitmaps
   Cursors

6. **What are the geometry manager classes of Tkinter ? (Understanding)**

   The three geometry manager classes of Tkinter .

   pack() method:It organizes the widgets in blocks before placing in the parent widget.
   grid() method:It organizes the widgets in grid (table-like structure) before placing in the parent widget.
   place() method:It organizes the widgets by placing them on specific positions directed by the programmer.

7. **How events are handled in python? (Analyzing)**

   Tkinter provides a powerful mechanism to deal with events. For each widget, we can bind Python functions and methods to events.

   widget.bind(event, handler)

   If an event matching the *event* description occurs in the widget, the given *handler* is called with an object describing the event.

8. **Capturing clicks in a window**

```
from Tkinter import *

root = Tk()

def callback(event):
print "clicked at", event.x, event.y

frame = Frame(root, width=100, height=100)
frame.bind("<Button-1>", callback)
frame.pack()

root.mainloop()
```

9. **Define Socket. (Remembering)**
   Sockets are the endpoints of a bidirectional communications channel. Sockets may communicate within a process, between processes on the same machine, or between processes on different continents.

10. **What are the methods of server socket? (Understanding)**

    - s.bind() - This method binds address (hostname, port number pair) to socket.
    - s.listen() - This method sets up and start TCP listener.
    - s.accept() - This passively accept TCP client connection, waiting until connection arrives

11. How sockets are created in python? (Applying)

Socket programming is started by importing the socket library and making a simple socket.

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Here we made a socket instance and passed it two parameters. The first parameter is **AF_INET** and the second one is **SOCK_STREAM**. AF_INET refers to the address family ipv4. The SOCK_STREAM means connection oriented TCP protocol.

12. How emails are sent using python? (Applying)

Simple Mail Transfer Protocol (SMTP) is a protocol, which handles sending e-mail and routing e-mail between mail servers.

Python provides **smtplib** module, which defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon.

Here is a simple syntax to create one SMTP object, which is used to send an e-mail −

```
import smtplib

smtpObj = smtplib.SMTP( [host [, port [, local_hostname]]]
```

13. What are modules/packages supported for sending emails in python? (Understanding)

- email
- rfc822
- smtplib
- mimetools
- mimetypes
- mailbox

14. Define CGI. (Remembering)

The Common Gateway Interface (CGI) is a standard for writing programs that can interact through a Web server with a client running a Web browser.

CGI is the standard for programs to interface with HTTP servers.
CGI programming is written dynamically generating webpages that respond to user input or webpages that interact with software on the server

15. Compare GET and POST method. (Evaluating)

GET requests can include a query string as part of the URL

GET method delivers data as part of URI

When using forms it's generally better to use POST:

POST method delivers data as the content of a request

<FORM METHOD=POST ACTION=…>

there are limits on the maximum size of a GET query string a post query string doesn't show up in the browser as part of the current URL

16. Give example for CGI script. (Understanding)

```
import cgi def main():
print "Content-type: text/html\n"
form = cgi.FieldStorage()          # parse query
if form.has_key("firstname") and form["firstname"].value != "":
        print "<h1>Hello", form["firstname"].value, "</h1>"
else:
        print "<h1>Error! Please enter first name.</h1>"
```

17. How to upload file using CGI in python? (Analyzing)

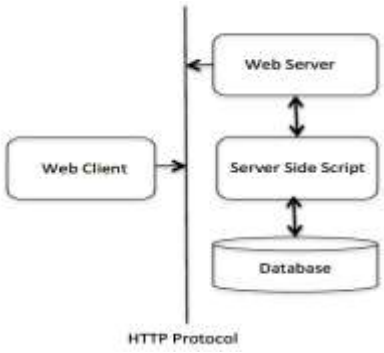
```
import cgi
form = cgi.FieldStorage()
if not form:
print """
<form action="/cgi-bin/test.py" method="POST" enctype="multipart/form-data">
<input type="file" name="filename">
<input type="submit">
</form>
"""
elif form.has_key("filename"):
item = form["filename"]
if item.file:
data = item.file.read()                    # read contents of file
print cgi.escape(data)
```

18. Draw the CGI architecture diagram. (Remembering)

Browser contacts the HTTP web server and demands for the URL

Web Server parses the URL and looks for the filename.

Web browser takes response from web server and displays either the received file or error message.



Part B (16 Marks)

1. Explain about Tkinter and write python script for calculator. **(Creating)**
2. Explain the steps to create widgets. **(Understanding)**
3. Briefly explain the different widgets of Tkinter. **(Remembering)**
4. Describe about socket programming in python. **(Remembering)**
5. Write a python code for CGI programming. **(Applying)**
6. Explain GET and POST method of CGI with an example. **(Remembering)**