TWO MARKS QUESTIONS WITH ANSWERS

1. Define Computer.

A computer is an electronic data processing device, which accepts and stores data input, processes the data input, and generates the output in a required format.

2. Define algorithm.

The sequence of steps to be performed in order to solve a problem by the computer isknown as an algorithm.

Programs = Algorithms + Data

3. What are the two phases in algorithmic problem solving?

The algorithmic problem solving actually comes in two phases:

- Derivation of an algorithm that solves the problem
- Conversion of the algorithm into program code

The first phase is the algorithmic phase and the second phase is the coding phase.

4. Why Algorithmic phase is the difficult phase? Justify

The algorithmic phase is the difficult phase, for two main reasons.

- Firstly, it challenges the mental facilities to search for the right solution.
- Secondly, it requires the ability to articulate the solution concisely into step by-step instructions, a skill that is acquired only through lots of practice.

5. What are the steps involved in algorithm development process?

The algorithm development process consists of five major steps.

Step 1: Obtain a description of the problem.

Step 2: Analyse the problem.

Step 3: Develop a high-level algorithm.

Step 4: Refine the algorithm by adding more detail.

Step 5: Review the algorithm.

5. Compare Computer Hardware and Software.

Description	*Hardware*	*Software*
Definition	Devices that are required to store and execute the software.	Computer Software is a set of instructions for a computer to perform specific operations.
Types	Input Devices, Output Devices, Storage Devices, Processing Devices, Control Devices.	System Software, Programming Software, Application Software.
Example	CD-ROM, Keyboard, Monitor, Printer, Scanner.	Adobe Acrobat, Microsoft Word
Nature	Hardware is physical in nature.	Hardware is logical in nature.

. State some of the properties of an algorithm.

- Finiteness
- Definiteness
- Effectiveness
- Input
- Output
- Feasibility
- Generality

What are the three building blocks in a Algorithm?

The three building blocks are instructions, state, control flow, and functions

Building Block	*Common name*
Instruction	Action/Sequence
State/Selection	Decision
Control Flow/Iteration	Repetition or Loop

9. **Define assignment, input and output statement.**
   - **Assignment:** Any instruction that assigns value to a variable
   - **Input:** Any instruction that gets the input values
   - **Output:** Any instruction that displays or prints the input values

10. **Describe different notations in algorithm and classify each.**
Algorithms can be expressed in many different notations, including Natural Language, pseudo code, flowcharts and programming languages.
   - Natural language tends to be verbose and ambiguous, and is rarely used.
   - Pseudo code and flowcharts are structured ways to express algorithms.
   - Programming languages are intended for expressing algorithms in a form that can be executed by a computer.

11. **Define Pseudocode.**

Pseudocode is an informal language used by programmers to develop algorithms. Pseudocode is a "text-based" detail design tool.

Example:

12. **Define some of the rules to be followed on Pseudocode.**

Rules to follow:

- Only one statement per line
- Indent to show hierarchy.
- Keep statement language independent.
- Capitalized initial keyword
- End multi-line structures.

13. **Define flowchart.**

A flowchart is a graphical representation of an algorithm. A flowchart is a diagram made up of boxes, diamonds and other shapes, connected by arrows - each shape represents a step in the process, and the arrows show the order in which they occur.

14. What are the different types of flowcharts?

- ✿ Document flowcharts
- ✿ Data flowcharts
- ✿ System flowcharts
- ✿ Program flowchart

15. What are the symbols used in flowcharting type?

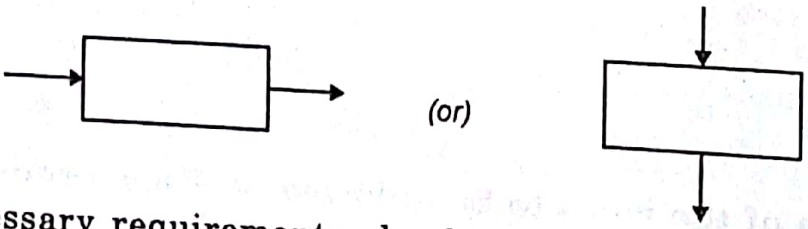
- ✿ Process / Operation Symbols
- ✿ Branching and Control of Flow Symbols
- ✿ Input and Output Symbols
- ✿ File and Information Storage Symbols
- ✿ Data Processing Symbols

16. Describe the Data Processing Symbols in Flowchart.

- ✿ Collate
- ✿ Sort

17. List some guidelines for drawing flowchart.

- ✿ The flowchart should be clear, neat and easy to follow.
- ✿ The usual direction of the flow of a process is from left to right or top to bottom.
- ✿ Only one flow line should come out from a process symbol.



- ✿ All necessary requirements should be listed out in logical order for drawing a proper flowchart.

18. List the merits in drawing the Flowcharts.

- ✿ Easy Communication
- ✿ Used for Effective analysis
- ✿ Proper documentation
- ✿ Efficient Coding
- ✿ Proper Debugging
- ✿ Efficient Program Maintenance

19. List the demerits in drawing the flowchart.

- ✿ Complex logic
- ✿ Alterations and Modifications
- ✿ Reproduction

20. Explain some of the qualities of Programming Language?

- Languages are not designed to provide a means for having a two-way dialog with a computer.
- It is a set of instructions specified by the human on what the computer should do.
- Provides a way for humans to communicate to computers.
- It always follows grammar; hence there is no ambiguity.
- A system will be able to infer the computer language.

21. What is the difference between the algorithm and the program?

A program is the implementation of an algorithm to be run on a specific computer and operating system. An algorithm is more abstract.

22. Give the Pseudocode to check the biggest of 2 numbers.

```
if (A>B)
      Print "A is greater"
else
      Print "B is greater"
```

23. State the differences between Iteration and Recursion.

Iteration	*Recursion*
Iteration explicitly uses a repetition structure.	Recursion achieves repetition through repeated function calls.
Iteration keeps modifying the counter until the loop condition fails.	Recursion keeps producing simple versions of the original problem until the base case is attained.
It reduces the processor's operating time.	It increases the processor's operating time.
Iteration normally occurs within the loop, hence no need of extra memory.	Recursion causes another copy of the function; hence considerable memory space is occupied.

REVIEW QUESTIONS

1. Explain the basic building blocks of algorithms with the neat sketch and examples
2. Explain the characteristics of Algorithms.
3. A word is a palindrome if it reads the same when the order of its character is reversed. For instance, "123321", "RADAR" and "step on no pets" ar palindromes. Derive an algorithm to determine if a word is a palindrome.
4. Illustrate with examples the different symbols of flowchart.
5. Two words are anagrams of each other if they differ only by the order o their characters. For example, "HATE" and "HEAT", "NOW" and "WON" "reset" and "steer" are anagrams, but "sell" and "less" are not. Write ar algorithm to determine if two given words are anagrams.
6. What are the guidelines for drawing flowcharts?
7. Give a suitable example for Iterative Algorithms.
8. Write the algorithm and flowchart for the following problem:
   In a mastermind game, the code maker hides a secret code consisting of 4 digits, and the code breaker is to provide 4-digit guess codes until he gets the code right. The code maker gives feedback in this manner, a sink is awarded to a perfect match of a digit plus its position, and a hit refers to a right match of digit but in the wrong position. How do you devise an algorithm to provide the correct feedback? First, work on the special case that no digit in the code should be repeated. Then work on the general case where digits can be duplicated.
9. Explain the concept of Recursion in detail. Explain Towers of Hanoi Problem.

TWO MARKS QUESTIONS WITH ANSWERS

1. What is a program?

A computer program is a collection of instructions that performs a specific task when executed by a computer.

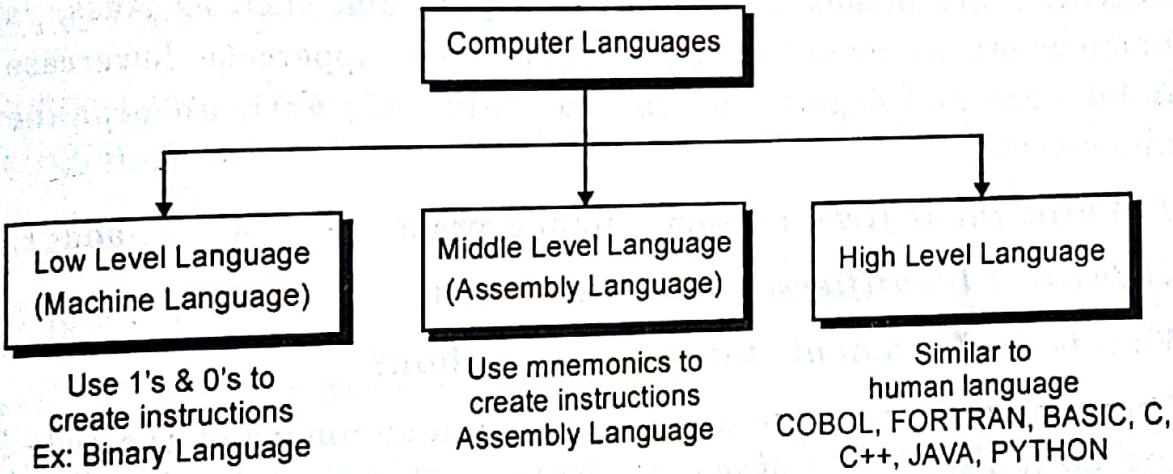
Example: Microsoft Word, Powerpoint, Excel etc.

2. Define Computer languages.

Computer languagesare the languages through which user can communicate with the computer by writing program instructions.

3. How Computer languages are classified?

Computer languages can be classified into low level, middle level and high level languages.



Types of Computer Languages

4. Define assignment statement.

The assignment statement is thata variable is always on the left hand side of an equal sign,and the right hand side it could be

- a value
- another variable
- an arithmetic expression

Syntax:

variable= expression

5. List the statements in python.

- Assignment statements
- Print statements

6. Given the strings, x = 'alpha' and y = 'beta' print the following string operations

print x + y= alphabeta

print y * 3 = betabetabeta

7. List some of the keywords in Python.

and	del	from	not	while	as	elif	global	or
with	assert	else	if	pass	yield	break	except	import
print	class	exec	raise	in	continue	is	finally	return
def	for	lambda	try	True	False	None		

8. Define identifiers in Python.

Identifiers are names for entities in a program, such as class, variables, functions etc. An identifier can be composed of uppercase, lowercase letters, underscore and digits, but should start only with an alphabet or an underscore.

i) Valid Identifiers : sum total average _ab_ add_1 x1

ii) Invalid Identifiers : 1x char x+y

9. What is the Comment statement in Python?

Comment statements are used to provide a summary of the code in plain English to help future developers better understand the code.

i) Single Line comments

#This is a commentstatement

ii) Multi-line comments

#This is a long comment

#and it extends

#to multiple lines

10. Define variables in Python.

Variables are nothing but reserved memory locations to store values. Values may be numbers, text or more complicated types. Declaration of variables is not required in Python. ***Example:*** addition = 900

11. Explain Input and Output statements in Python.

i) Input Statement:

- Python provides two built-in functions to read a line of text from standard input device, keyboard. These functions are
  - input
  - raw_input

ii) Output Statement

- print() function is used to output data to the standard output device (screen). print statement is used where zero or more expressions are passed separated by commas.
  - print "Python is really a great language!!!"

12. List the Data types in Python.

In Python, everything is represented as an object and every object has an identity, a type, and a value. Python has five standard data types:

- Numbers
- String
- List
- Tuple
- Set
- Dictionary

13. What are the operators in Python?

Python language supports the following types of operators.

- Arithmetic Operators
- Comparison Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators
- Membership Operators

14. Solve the mathematical expression 7 / 3 * 1.2 + 3 / 2."

7/3 *1.2+3/2

2 * 1.2 + 3 / 2

2.4 + 3 / 2

2.4 + 1 "

15. Define Functions in Python.

Functions are common to all programming languages. It is defined as a bloc of related statements to perform a specific task. Functions help programmer to break the program into small manageable units or modules.

Syntax of Function

```
def function_name (parameters)
    """docstring"""
    statement(s)
return
```

16. Explain the precedence of operators in Python.

The following table lists precedence of operators with highest precedence at top and lowest precedence at bottom. Operators in the same cell evaluate from left to right.

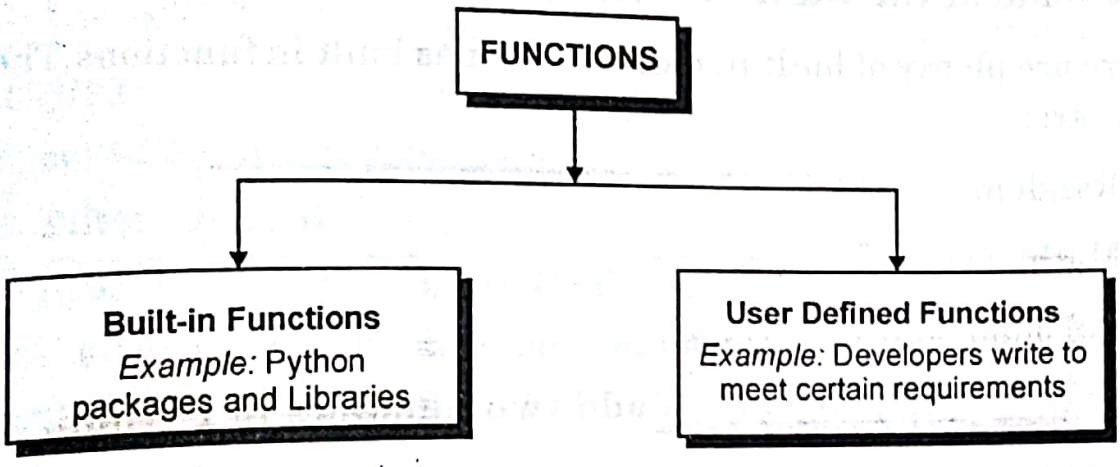
Operator	*Description*
**	Exponentiation
~ + –	Complement, unary plus and minus
*, /, %	Multiplication, division, remainder
+, –	Addition, subtraction
<<, >>	Bitwise shifts
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
in, not in, is, is not, <, =, >, >=,<>, !=, ==	Comparisons, membership, identity
not x	Boolean NOT
and	Boolean AND
or	Boolean OR
lambda	Lambda expression

17. What are the Function arguments used in Python?

Following types of formal arguments are used in python:

- Required arguments
- Keyword arguments
- Default arguments
- Variable-length argument or Arbitrary argument

18. List the types of functions.



Types of Functions in Python

19. What is the Anonymous Function in Python?

Anonymous functions are the functions that are not defined like normal functions by using the keyword def. In Python anonymous functions are defined using the keyword lambda.

Syntax: lambda arguments: expression

20. Define Modules in Python

Module is a file containing Python definitions and statements. It defines functions, classes and variables and includes runnable code also. Functions are groups of code and modules are groups of functions.

21. How modules are incorporated in a Python program?

Modules are incorporated in the Python program using the from...import statement.

- For a module involving hundreds of functions, from.....import Statement recommended to save loading time.

  Syntax: *from python_file import function_name*

- Multiple functions can be imported by separating their names with comma

  Syntax: *from python_file import_function_name1, function_name 2*

- Use the asterisk symbol(*) to import everything

  Syntax: *from python_file import**

22. List some of the Built-in Modules in Python

There are plenty of built in modules, just as built in functions.The most use ones are:

- Random
- Math
- Calendar and date-time

23. Write a Simple program to add two numbers in Python.

Program

```
1   # This program adds two numbers
2
3   num1 = 1.5
4   num2 = 6.3
5
6   # Add two numbers
7   sum = num1 + num2
8
9   # Display the sum
10  print 'Sum = ',  sum
```

EXECUTION

```
sh-4.3$ python program.py
Sum =  7.8
```

24. Write a Simple program to convert KM/H to MPH in Python.

This program converts speed from KM/H to MPH, which may be handy for calculatingspeed limits when driving abroad, especially for UK and US drivers.

Formula to convert KM/H to MPHis : 1 kilometre = 0.621371192 miles

Program

```
1   kmh = int(raw_input("Enter km/h: "))
2   mph =  0.6214 * kmh
3   print "Speed:", kmh, "KM/H = ", mph, "MPH"
```

EXECUTION

```
sh-4.3$ python program.py
Enter km/h: 5
Speed: 5 KM/H =  3.107 MPH
```

25. Write a Simple program to convert decimal number into binary, octal and hexadecimal number system in Python.

Program

```
1   # Python program to convert decimal number into binary,
octal and hexadecimal number system
2   dec = 344
3   print "344 in Binary =", bin(dec)
4   print "344 in Octal =", oct(dec)
5   print "344 in Hexadecimal =",hex(dec)
```

EXECUTION

```
sh-4.3$ python program.py
344 in Binary = 0b101011000
344 in Octal = 0530
344 in Hexadecimal = 0x158
```

Note: In this program, we have used built-in functions bin(), oct() and hex() to convert the given decimal number into respective number systems.

REVIEW QUESTIONS

1. Define Variables. What re the rules for defining Identifiers?
2. Explain Python Comment statement with suitable examples.
3. Explain Input statement in Python. What are the types of input?
4. Write a Python program to print the following string in a specific format:

Twinkle, twinkle, little star,
How I wonder what you are!
Up above the world so high,
Like a diamond in the sky.
Twinkle, twinkle, little star,
How I wonder what you are.

5. What are the data types in python? Explain each data type with suitable examples.
6. What are the types of operators supported by Python language? List the operators and explain them.
7. Define Function. Explain the scope and lifetime of the variables with suitable examples.
8. Explain the various function arguments in detail.
9. Explain Python modules in detail. Explain some of the built-in modules available in Python.

TWO MARKS QUESTIONS WITH ANSWERS

1. Define boolean datatype.

The Boolean data type is a data type, having two values (usually denoted true and false). It is named after George Boole, who first defined an algebraic system of logic in the mid 19thcentury.

2. What are the Conditional statements used in Python?

- if statements
- if...else statements
- if...elif...else statements(or) chained conditionals
- Nested if statements

3. Define if...else statements with its syntax.

The if..else statement evaluates test condition and executes the true statement block only when test condition is True.If the condition is False, false statement block is executed. Indentation is used to separate the blocks.

Syntax:

```
if condition:
    True Statement Block
else:
    False Statement Block
```

4. Write the syntax for ternary operator.

The syntax for ternary operaor is,

Syntax 1:

```
value_if_true_(if_condition) else value_if_false
```

Syntax 2:

```
(value_if_true, value_if_false) [if_condition]
```

5. Define chained conditionals.

The elif is short form for else if. It is used to check multiple conditions. If the condition 1 is false, it checks the condition 2 of the next elif block and so on.If all the conditions are False, then the else statement is executed. The if block can have only one else block. But it can have multiple elif blocks

6. Write the syntax for if...elif...else conditionals.

Syntax:

if condition 1:

True Statement Block for Condition 1

elif condition 2:

True Statement Block for Condition 2

elif condition 3:

True Statement Block for Condition 3

else

False Statement Block

7. Define iteration.

Iteration is otherwise called as looping.There are situations wh programmers need to execute a block of code several number of time Repeated execution of a set of statements is called iteration or looping.

8. What are the different iterative statements?

Python programming language provides following types of iterati statements.

- ✿ The for loop
- ✿ The while statement
- ✿ Nested loops

9. Define range() function and its syntax.

Sequence of numbers are also generated using range() function.

Syntax:

range(start_element, stop_element, step size)

Default step size is equal to 1 if not provided.

10. Define the While loop.

The while loop in Python iterates over a block of statements as long as th test condition is true.The statement body is entered only when the tes condition is true. The process continues until the test condition evaluate to False.

Syntax:

```
while condition:
    statement_1
    ...
    statement_n
```

11. Write the syntax for Nested for loops and Nested while loop statements.

Python programming language allows using one loop inside another loop.

Syntax:

```
Nested for loop
for iterating_var in sequence:
    statement(s)
    statement(s)
```

Syntax:

```
Nested while loop
while expression:
while expression:
    statement(s)
    statement(s)
```

12. What is Python Break statement?

The break statement in Python terminates the current loop and resumes execution at the next statement. It's just like the traditional break found in C.

Syntax: break;

13. What is Python Continue statement?

The continue statement moves the control to the beginning of the while or for loop. The continue statement rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop.

Syntax: continue;

14. Define Fruitful function in Python.

A function in Python

- Takes input data, called parameters or arguments.
- Performs some computations.
- Returns the result.

15. What are types of parameters in Functions?

Parameter is the input data that is sent from one function to another. T parameters are of two types:

- ***Formal parameters***
  - This parameter defined as part of the function definition.
  - The actual parameter value is received by the formal parameter.
- ***Actual parameters***
  - This parameter defined in the function call.

16. What are thevarious Parameter Passing Techniques?

There are two types of parameter passing in programming languages.

Call by value: In call by value, a copy of actual arguments is passed formal arguments and any changes made to the formal arguments have effect on the actual arguments.

Call by reference: In call by reference, the address of actual arguments passed to formal arguments. By accessing the addresses of formal argument it will be reflected in the actual arguments too.

17. Write the Scope of the Variable.

A variable in the program can be either local variable or global variable. global variable is a variable that is declared in the main program while local variable is a variable that is declared within the function.

```
s = 10 <---- Here, s is the Global variable
def f1():
    s= 55 <---- Here, s is the Local variable
    print s      # output = 55
```

18. What is Recursive Function and its limitations?

In Python, a function is recursive if it calls itself and has a termination condition.

Limitations of recursions

Every time a function calls itself and stores some memory. Thus, a recursive function could hold much more memory than a traditional function.

19. Write the merits of using Functions in a program.

- Dividing a large program into functions allow the programmers to read and debug easily
- A function allows reusing code instead of rewriting it.
- A function allows testing small parts of program in isolation from the rest.

20. Write the syntax for Composition.

When a function is called from within another function, it's called composition.

Syntax:

```
def outer();
def inner(a);
    return a
return inner
```

21. Define strings and name some methods.

A string is a sequence of characters ie. they can be letter, a number, or a backslash. Python strings are "immutable" which means they cannot be changed after they are created.

22. List some of the methods in List Operations.

Length, Concatenation, Repetition, Append, Insert, Extend, Remove, Pop, Clear, Index, Count, Sort and Reverse.

23. State the differences between Linear search and Binary search.

Linear search, also known as the sequential search is the simplest search algorithm. It searches for a specified value in a list by checking every element in the list.

Binary search is also a method used to locate a specified value in a sorted list. Binary search method halves the number of elements checked thereby reducing the time taken.

REVIEW QUESTIONS

1. Explain If structure in detail. Explain with suitable examples.
2. Explain iteration structure in detail with suitable examples.
3. Write a Python program that accepts a string and calculate the number digits and letters.

   Given String: Python 3.2

   Expected Output:

   Number of letters : 6

   Number of digits: 2
4. Write a Python program to display the current date and time.
5. Explain Python Break, Continue and Pass Statements with examples.
6. Write a Python program that prints all the numbers from 0 to 6 except 3 and
7. What is a fruitful function? What are the parameters used in the fruit functions?
8. What are the two parameter passing techniques? Explain them wi examples.
9. Write a Python function that accepts a string and calculate the number upper case letters and lower case letters.

   Sample String :Nothing WORTH comes EASY.

   Expected Output :

   No. of Upper case characters :10

   No. of Lower case Characters : 11

TWO MARKS QUESTIONS WITH ANSWERS

1. Define List.

A list is a data structure in Python that is a mutable, or changeable, ordered sequence of elements. Each element or value that is inside of a list is called an item. Each one of them is numbered, starting from zero - the first one is numbered zero, the second 1, the third 2, etc. You can remove values from the list, and add new values to the end

2. What is Cloning of List?

Cloning is the process of modifying a list and also keeps a copy of the original. The entire list is copied not just the reference. The easiest way to clone a list is to use the slice operator.

Example:

```
a = [ 5, 10, 50, 100]
b = a[0:2]
b= [5, 10]
```

3. What is Aliasing?

More than one list variable can point to the same data. This is called an Alias.

For example:

```
a = [ 5, 10, 50, 100]
b = a
```

Now, a and b both point to the same list.

Without the slice operator, a and b point to the same list. Changes to one affect the other. With it, they point to different copies of the list that can be changed independently.

4. Define Tuple.

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

Explain Tuple Assignment with example.

Python has a very powerful tuple assignment feature that allows a tuple of variables on the left of an assignment to be assigned values from a tuple on the right of the assignment.

(name, surname, b_year, movie, m_year, profession, b_place) = julia

This does the equivalent of seven assignment statements, all on one easy line. One requirement is that the number of variables on the left must match the number of elements in the tuple.

6. **What is Slicing?**

A Python slice extracts elements, based on a start and stop. We specify an optional first index, an optional last index, and an optional step. For Example;

```
values = [100, 200, 300, 400, 500]
# Get elements from second index to Third index.
slice = values[1:3]
# Slice from third index to index one from last.
slice = values[2:-1]
# Slice from start to second index.
slice = values[:2]
# Slice from second index to end.
slice = values[2:]
```

7. **Define Dictionary.**

Python dictionary is an unordered collection of items. While other compound data types have only value as an element, a dictionary has a key: value pair. Dictionaries are optimized to retrieve values when the key is known.

8. **Give an example for List Comprehension**

The concept of "list comprehensions" is used to construct lists in an easy and natural way. List comprehensions can also create a list with a sub-set of elements from another list by applying a condition. The list comprehensions make coding simpler and efficient; executes much faster than FOR loop.

For Example;

EvenNos = [n for n in range(20)if n % 2 == 0]=>[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]

9. What is Mutability?

A **mutable object** can be changed after it's created, and an immutable object can't. Lists are mutable and tuples are immutable.

10. List the functions of Tuple Data Type.

S.No.	*Function*	*Description*
1.	cmp(tuple1, tuple2)	Compares elements of both tuples.
2.	len(tuple)	Gives the total length of the tuple.
3.	max(tuple)	Returns item from the tuple with max value.
4.	min(tuple)	Returns item from the tuple with min value.
5.	tuple(seq)	Converts a sequence into tuple.
6.	sum(tuple, S)	Return the sum of all elements and S
7.	sorted(tuple)	Sorts the tuple in ascending or descending order

11. List the methods of List Data Type.

S. No.	*Method*	*Description*
1.	append(E)	Add an element(E) to the end of the list
2.	extend(seq)	Add all elements of a list to the another list(seq)
3.	insert(ind,E)	Insert an element (E) at the defined index(ind)
4.	remove(E)	Removes the first occurrence of element (E) from the list
5.	pop(ind)	Removes and returns an element at the given index (ind)
6.	index(E)	Returns the index of the first matched element with E
7.	count(E)	Returns the count of number of elements (E) in the list
8.	sort()	Sort items in a list in ascending order
9.	reverse()	Reverse the order of items in the list
10.	copy()	Returns a shallow copy of the list

12. **Comment on Tuple as Return type.**

In general a functions can always only return a single value. However when the return datatype is a tuple we can combine together as many values as we like, and return them together.

For example, we could write a function that returns both the perimeter and the area of a square:

```
defArea_Perimeter(Side):
    Area = Side * Side
    Perimeter = 4 * Side
    return (Perimeter, Area)
```

13. **When a dictionary does is used instead of a list?**

Dictionaries - are best suited when the data is labelled, i.e., the data is a record with field names.

Lists - are better option to store collections of un-labelled items say all the files and sub directories in a folder.

Generally Search operation on dictionary object is faster than searching a list object.

14. **Differentiate between append() and extend() methods. ?**

Both append() and extend() methods are the methods of list. These methods are used to add the elements at the end of the list.

append(element) - adds the given element at the end of the list which has called this method.

extend(another-list) - adds the elements of another-list at the end of the list which is called the extend method.

15. **What is the output of print list + tinylist * 2 if list = ['abcd', 786 , 2.23, 'john', 70.2] and tinylist = [123, 'john']?**

It will print concatenated lists. Output would be ['abcd', 786, 2.23, 'john', 70.2, 123, 'john', 123, 'john'].

16. What is the difference between tuples and lists in Python?

The main differences between lists and tuples are - Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as read-only lists.

17. What is the difference between del() and remove() methods of list?

To remove a list element, you can use either the del statement if you know exactly which element(s) you are deleting or the remove() method if you do not know.

18. How to merge two dictionaries?

The update method update() merges the keys and values of one dictionary into another, overwriting values of the same key

```
Dict1 = {"house":453,"Pet":"Cat","Color":"Brown"}
Dict2 = {"Age":2","Color":"white"}
w.update(w1)
print w
{"house":453,"Pet":"Cat","Age":2","Color":"white"}
```

. Define mutable and immutable data type.

Immutable data value: A data value which cannot be modified. Assignments to elements or slices (sub-parts) of immutable values cause a runtime error.

Mutable data value: A data value which can be modified. The types of all mutable values are compound types. Lists and dictionaries are mutable; strings and tuples are not.

When is a dictionary used instead of a list?

Dictionaries - are best suited when the data is labelled, i.e., the data is a record with field names.

Lists - are better option to store collections of un-labelled items say all the files and sub directories in a folder.

Generally Search operation on dictionary object is faster than searching a ist object.

REVIEW QUESTIONS

Illustrate with suitable examples the methods of Lists.

Explain in detail List data type.

Explain with examples Aliasing and Cloning in List.

Explain in detail List Slicing

Explain the data type dictionary.

Write a Python script to generate and print a dictionary that contains a number (between 1 and n) in the form (x, x*x).

What are tuples? Explain in detail the immutable property of tuple.

Write a Python program to find number of times each element is present in the tuple.

Example: tuplex = 2, 4, 5, 5, 2, 5, 4, 4, 5

Output:

Number 2 - 2 times

Number 4 - 3 times

Number 5 - 4 times

With example programs illustrate List Comprehensions.

TWO MARKS QUESTIONS WITH ANSWERS

1. Define File.

A file refers to a location with filename that stores information. The storage area is non-volatile memory like hard-disk.A file stores related data, information, settings, or commands in secondary storage device like magnetic disks, magnetic tapes and optical disks.A file can be a sequence of bits, bytes, lines or records depending of the application/software used to create it. For example a text file is organized as a sequence of lines.

2. List the File Opening Modes.

Modes	*Description*
r	Opens a file for reading only.
r+	Opens a file for both reading and writing.
w	Opens a file for writing only. Overwrites the file if the file exists.
w+	Opens a file for both writing and reading.
a	Opens a file for appending. File pointer is at the end of the file.
a+	Opens a file for both appending and reading.

3. List the different ways to read a file.

The text files can be read in four different ways listed below

- Using Read Method
- Using ReadlinesMethod
- Using For Line In File Method
- Using Readline Method

4. What is the difference between append and write mode?

The write pointer is set to end of file when a file is opened in "a" mode. In append mode the file can never be read. The contents can only be written at the end of the file. In the write mode the write pointer is set to the beginning of the file.

5. What are the attributes of file objects?

Attribute	*Description*
file.closed	If file is closed returns true else false
file.mode	Returns one of the mode listed in table 5.2
file.name	Returns name of the file.
file.softspace	Returns false if space explicitly required with print, true otherwis

6. List the methods in file objects.

S.No	*Method*	*Description*
1.	close()	Close an opened file.
2..	read(n)	Reads at most n characters from the file.
3.	readable()	Returns True if the file stream can be read fr
4.	readline(n=-1)	Read and return one line (at most n bytes) fro file.
5.	readlines(n=-1)	Read and return a list of lines (at most n bytes) the file.
6.	seek(offset, from = SEEK_SET)	Change the file position to offset bytes, in refer to from
7.	seekable()	Returns True if the file stream supports ran access.
8.	tell()	Returns the current file location.
9.	writable()	Returns True if the file stream can be written t
10.	write(s)	Write string s to the file and return the numbe characters written.
11.	writelines(lines)	Write a list of lines to the file.
12.	flush()	Flushes the internal buffer
13.	fileno()	Returns an integer which is the file descript Depends on the underlying operating system.
14.	isatty()	Returns true if the file is connected to any termin device
15.	next()	Returns the next line from the file each time it i being called.
16.	truncate(n)	The file is truncated to at most n bytes

7. Differentiate Errors and Exceptions.

Errors are caused by the mistakes in the program. There are three types of errors; Syntax errors, Runtime Errors and Logical Errors. The syntax errors can be rectified when the complier throws errors. The logical errors are erroneous output; the program gets executed but the output is not the desired one. A syntactically correct statement might cause errors during execution. Errors detected during execution/runtime are called exceptions.

8. Illustrate try-except-else.

```
1.  # Example for try-except-else
2.
3.  try:
4.  file_read= open("test.txt", "r")
5.      X = file_read.read()
6.      X = X+8
7.  except IOError:
8.          print "Error: can\'t find file or read data"
9.  except :
10.         print "Some other error"
11. else:
12.         print "Content of file", X
13. finally:
14. file_read.close()
15.        print "Finally Close the file"
```

EXECUTION 1:

When test.txt exists; error in line 4: cannot add string with integer

```
sh-4.3$ python main.py
Some other error
Finally Close the file
```

EXECUTION 2:

When test.txt doesn't exist ; file is not opened and hence closing in 12 is error

```
sh-4.3$ python main.py
Error: can't find file or read data
Traceback (most recent call last):
  File "main.py", line 12, in <module>
file_read.close()
NameError: name 'file_read' is not defined
```

9. Define Modules.

In Python, module is the way to structure program. Each Python program file is a module, which imports other modules like objects and attributes.

10. Define Packages.

A package is a collection of modules in directories that give a package hierarchy. When a complex application/program is created it is better to be organized. The package gives the hierarchical file directory structure of the Python application environment that consists of modules and subpackages and sub-subpackages, and so on.

11. Define Pickling.

Pickle module accepts any Python object and converts it into a string representation and dumps it into a file by using dump function, this process is called pickling. While the process of retrieving original Python objects from the stored string representation is called unpickling.

12. Give the mechanism to handle exceptions.

The try and except statements are used to handle the runtime errors. The syntax for normal try-except block is given below

```
try:
    # lines of code that might encounter runtime error
except:
    # lines of code that will be executed when runtime error occurs
```

In the below program; line 6 will never be executed if the line 4 is absent. Only when an exception happens, the except block would be executed. Moreover the lines after the error line will never be executed in the try block; note line 5 was not executed.

```
1.  # Divide by Zero exception
2.  try:
3.        print "Hello World"
4.        x= 10/0
5.        print "Never Executed"
6.  except:
7.  print "This is an error message!"
```

EXECUTION

```
sh-4.3$ python main.py
This is an error message!
```

13. How to raise an exception?

The raise statement in python is used to forcefully invoke an exception. The syntax for raising an exception is

raise<ExceptionName>

theExceptionName can the exceptions listed in table 5.6 or some used defined exception.

14. What is __init__.py used for?

It declares that the given directory is a package. When a file _init_.py i placed in a folder XYZ; it means that folder XYZ is a package.

REVIEW QUESTIONS

1. Illustrate with suitable examples various modes of opening Files.
2. Explain the difference between append mode and write mode.
3. Write a python program to find the longest word in a file.
4. Differentiate Errors and Exceptions with suitable examples.
5. Explain the different try clauses.
6. Explain with examples Modules and Packages.
7. Explain in detail the format operator.

Model Question Papers

B.E./ B.TECH DEGREE EXAMINATION

First Semester

GE 8151 - PROBLEM SOLVING AND PYTHON PROGRAMMING

(Common to All Branches)

(Regulation 2017)

Time : Three Hours **Maximum : 100 Marks**

Answer All Questions

PART - A (10 × 2 = 20 Marks)

1. Define some of the rules to be followed on Pseudo code.
2. What is iteration? Is iteration the same as repetition?
3. Define identifiers in Python
4. What is the Anonymous Function in Python?
5. How do you generate random numbers in Python?
6. Write a function that calculates the average of a list of numbers.
7. What is a nested list? Does the slice operator always produce a new list?
8. How do you find how big a dictionary is?
9. What's the difference between readline, readlines, read, write and writelines?
10. Describe what options you have to specify the format of numbers in Python.

PART - B (5 × 16 = 80 Marks)

11. a) i) Draw the flowchart for finding the biggest number from the three numbers.

    ii) What are the guidelines for drawing flowcharts?

    (OR)

    b) i) Explain the concept of Recursion in detail.

    ii) Write an algorithm to find the factorial of n numbers using recursion.

12. a) i) Explain Input statement in Python. What are the types of input?

    ii) Write a function that sorts an array of integers in ascending order.

(OR)

b) Explain Python modules in detail. Explain some of the built-in modules available in Python.

13. a) i) What are conditional statements and what can they do for us? Explain If structure in detail.

ii) Can you find the largest of two numbers without using an IF statement? What if the two numbers are integers?

(OR)

b) Explain iteration structure in detail with suitable examples.

14. a) What is a dictionary? Are dictionaries sequences? Is there any such thing as the empty dictionary? How can you add, delete, modify entries in a dictionary? In what circumstances are dictionaries useful? Define the terms key, and key-value pairs.

(OR)

b) i) What are tuples? Explain in detail the immutable property of tuple.

ii) Write a python program to sort a list of Nested Dictionaries.

5. a) i) Write a python programs to calculate:

- the number of lines in a file
- the number of words in a file
- the number of characters in a file

ii) Explain in detail the format operator.

(OR)

b) i) Explain the difference between append mode and write mode.

ii) What is pickling? Write a program that illustrates the notion.

B.E./ B.TECH DEGREE EXAMINATION

First Semester

GE 8151 - PROBLEM SOLVING AND PYTHON PROGRAMMING

(Common to All Branches)

(Regulation 2017)

Time : Three Hours **Maximum : 100 Marks**

Answer All Questions

PART - A (10 × 2 = 20 Marks)

1. Define a flowchart. Why a flowchart is required?
2. State the differences between Iteration and Recursion.
3. How 1,000 are printed in Python?
4. How can you swap the values of two variables in Python?
5. What is a local variable? State the differences between local and global variable.
6. What is the benefit and purpose of the string module?
7. How can you list the entries in a dictionary in alphabetical order?
8. Is a dictionary suitable for storing username-password pairs?
9. What are exceptions? What are they good for?
10. Can Python I/O handle directories?

PART - B (5 × 16 = 80 Marks)

11. a) Explain Towers of Hanoi Problem with a neat sketch. Write the algorithm to explain the Tower of Hanoi problem in detail.

(OR)

b) i) Write the pseudo code and algorithm to find whether the given number is odd or even.

ii) Give a suitable example for Iterative Algorithms.

12. a) i) Write a python program that accepts the base and height of the triangle and compute the area.

ii) Write a python program to convert kilometres to Miles.

(OR)

b) i) Define Function. Explain the scope and lifetime of the variables with suitable examples.

ii) Explain the various function arguments in detail.

13. a) i) Write a program that determines the frequency of occurrence of vowels in a string.

ii) Write a program that reverses the order of words in a sentence.

(OR)

b) i) What are the two parameter passing techniques? Explain them with examples.

ii) Write a Python program to display the current date and time.

14. a) i) Write a function that receives a list and a value and counts the number of occurrences of the value in the given list.

ii) Give examples of extend, reverse and append in a list context.

(OR)

b) With example programs illustrate List Comprehensions

15. a) i) Write a python programs to calculate:

- the average number of characters per line in a file
- the average number of words per line in a file
- the average number of characters per word in a file

ii) Differentiate Errors and Exceptions with suitable examples.

(OR)

b) Explain with examples Modules and Packages.

2.10 ILLUSTRATIVE PROGRAMS

2.10.1 Program to Swap Two Variables using Functions

PROGRAM 35

```
#Swap two variables using functions
def swap(x,y):
    "This function for swapping"
x,y=y,x                              # swap two variables
    print("After Swapping")
    print(x,y)

x = 10
y = 20
print("Before Swapping")
print(x, y)
swap(x,y)
```

EXECUTION

```
sh-4.3$ python program35.py
Before Swapping
(10, 20)
After Swapping
(20, 10)
```

2.10.2 Program to Find Fibonacci Series

PROGRAM 36

```
Nth number of Fibonacci Series using recursive function
def fib(n):
    if n == 0:
        return 0
elif n == 1:
        return 1
    else:
        return fib(n-1) + fib(n-2)

print 'Fibonacci number in 10th place is',fib(10)
```

EXECUTION

```
sh-4.3$ python program36.py
Fibonacci number in 10th place is 55
```

2.10.3 Program to Check Whether the Given Year is Leap Year or No

PROGRAM 37

```
1   # check whether the given year is leap year or not
2
3   year = int(input("Input the year = "))
4   if year % 4 == 0 and year %100 != 0 or year % 400 == 0
5       print ("\nGiven year is leap year")
6   else:
7       print ("\nGiven year is not leap year")
```

EXECUTION

```
sh-4.3$ python program37.py
Input the year = 2004
Given year is leap year
Input the year = 2002
Given year is not leap year
```

2.10.4 Program to Describe Usage of Global Variables

PROGRAM 38

```
1   #Usage of Global Variables
2   def sample():
3   return total + 100
4
5   total = 0                    # Global variable
6   print(sample())
```

EXECUTION

```
sh-4.3$ python program38.py
100
```

It is not possible to change the value of a global variable without explicitly specifying it.

```
1  def sample():
2  total = total + 100
3  print total
4  return total
5
6  total = 0
7  print(sample())
```

```
sh-4.3$ python program23.py
UnboundLocalError: local variable 'total' referenced before assignment
```

2.10.5 Program to Circulate the Values of n Variables

PROGRAM 39

```
1   # List Program to circulate values of n variables
2   def rotate(l, n):
3       new_list = l[n:] + l[:n]
4       return new_list
5
6   example_list = [1, 2, 3, 4, 5]
7   print "Original List = ", example_list
8   my_list = rotate(example_list, 1)
9   print "List rotated clockwise by 1 = ", my_list
10  my_list = rotate(example_list, 2)
11  print "List rotated clockwise by 2 = ", my_list
12  my_list = rotate(example_list, - 2)
13  print "List rotated anti-clockwise by 2 = ", my_list
```

ECUTION

```
sh-4.3$ python program39.py
Original List =  [1, 2, 3, 4, 5]
List rotated clockwise by 1 =  [2, 3, 4, 5, 1]
List rotated clockwise by 2 =  [3, 4, 5, 1, 2]
List rotated anti-clockwise by 2 =  [4, 5, 1, 2, 3]
```

2.10.6 Program to Display all the Prime Numbers within an Interval

PROGRAM 40

```
# Python program to display all the prime numbers
                                        within an interval
lower = input("Enter lower range: ")
upper = input("Enter upper range: ")
print "Prime numbers between",lower,"and",upper,"are:"

for num in range(lower,upper + 1):
   # prime numbers are greater than 1
   if num > 1:
       for i in range(2,num):
           if (num % i) == 0:
               break
       else:
print(num)
```

EXECUTION

```
sh-4.3$ python program40.py
Enter lower range: 1
Enter upper range: 20
Prime numbers between 1 and 20 are:
2
3
5
7
11
13
17
19
```

2.10.7 Program to Convert Temperature in Celsius to Fahrenheit

PROGRAM 41

```
1   # Python Program to convert temperature in celsius to
                                                    fahrenheit
2   celsius = 37.5
3   # calculate fahrenheit
4   fahrenheit = (celsius * 1.8) + 32
5   print('%0.1f degree Celsius is equal to %0.1f degree
                        Fahrenheit' %(celsius,fahrenheit))
```

EXECUTION

```
sh-4.3$ python program41.py
37.5 degree Celsius is equal to 99.5 degree Fahrenheit
```

2.10.8 Python Program to Check Armstrong Number

PROGRAM 42

```
1   # Python program to check Armstrong number
2   # An Armstrong number of three digits integer
3   # such that the sum of the cubes of its digits is
                                    equal to the number itself.
4   # For example, 371 is an Armstrong number
5   # 3**3 + 7**3 + 1**3 = 371.
6
7   num = int(input("Enter a number: "))
8   sum = 0
9
10  # find the sum of the cube of each digit
11  temp = num
12  while temp > 0:
13      digit = temp % 10
14      sum += digit ** 3
15      temp //= 10
16
17  # display the result
18  if num == sum:
```

```
19      print(num,"is an Armstrong number")
20  else:
21  print(num,"is not an Armstrong number")
```

EXECUTION

```
sh-4.3$ python program42.py
Enter a number: 663
(663, 'is not an Armstrong number')
Enter a number: 407
(407, 'is an Armstrong number')
```

10.9 Python Program to find the Distance between Two Points

PROGRAM 43

Distance Formula and Pythagorean Theorem

The distance formula is derived from the Pythagorean theorem. To find the distance between two points (x_1, y_1) and (x_2, y_2), the following formula is used.

$$\text{Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

```
1.  import math
2.
3.  p1 = [2 , 4]    # Point p1 co-ordinates
4.  p2 = [3 , 6]    # Point p2 co-ordinates
5.  distance = math.sqrt(((p2[0]-p1[0])**2)
                    +((p2[1]-p1[1])**2))
6.
7.  print " Distance between two points ", distance
```

EXECUTION

```
sh-4.3$ python program42.py
Distance between two points 2.2360679775
```

ILLUSTRATIVE PROGRAMS

.1 Program to Find the Square Root of the Given Number

ROGRAM 38

```
38  # Program to find the square root of the given number
39
40  import math    # This will import math module
41
42  print "Square of 49 = ", math.sqrt(49)
43  num = input("Enter the number :")
44  print "Square root of the given number = ", int(math.sqrt(num))
```

CUTION

```
sh-4.3$ python program38.py
Square of 49 =  7.0
Enter the number :49
Square root of the given number =  7
```

Program to Find GCD of Two Numbers

GRAM 39

```
11  #GCD of two numbers
12  #Divisors of 54 are: {1,2,3,6,9,18,27,54}
13  #Divisors of 24 are: {1,2,3,4,6,8,12,24}
14  #Common divisors of 54 and 24: {1,2,3,6}
15  #Greatest common divisor of 54 and 24 = 6
16
17  d1=int(input("Enter a number = "))
18  d2=int(input("Enter another number = "))
19  rem=d1%d2
20  while rem!=0 :
21      d1=d2
22      d2=rem
23      rem=d1%d2
24  print "GCD of given numbers is = ", d2
```

EXECUTION

```
sh-4.3$ python program39.py
Enter a number = 54
Enter another number = 24
GCD of given numbers is =  6
```

3.8.3 Program to Find Exponentiation of a Given Number

PROGRAM 40

```
1  # Exponentiation of a given number
2  import math
3
4  print "Exponentiation of a negative number = ",
                                    math.exp(-45.17)
5  print "Exponentiation of a negative number =  ", math.exp(10)
6  print "Exponentiation of a pi =   ", math.exp(math.pi)
```

EXECUTION

```
sh-4.3$ python program40.py
Exponentiation of a negative number =  2.41500621326e-20
Exponentiation of a negative number =   22026.4657948
Exponentiation of a pi =  23.1406926328
```

3.8.4 Program to Explain Enumerate() Function

PROGRAM 41

```
1  # Enumerate() function is used to iterate through a list
2  # while keeping track of the list items indices.
3
4  fruits = ['apples', 'bananas', 'blueberries',
                                   'oranges', 'mangos']
5
6  for index, fruit in enumerate(fruits):
7       print("The fruit, " +fruit + ", is in position "
                                   + str(index) + ".")
```

EXECUTION

```
sh-4.3$ python program41.py
The fruit, apples, is in position 0.
The fruit, bananas, is in position 1.
The fruit, blueberries, is in position 2.
The fruit, oranges, is in position 3.
The fruit, mangos, is in position 4.
```

3.8.5 Program to Find Sum of Elements in the List

PROGRAM 42

```
1   #Sum of elements in the list
2   my_data = [89,221,8,23]
3   sum = 0
4   for i in my_data:
5     sum += i
6   print "Sum of elements in the list = ", sum
```

EXECUTION

```
sh-4.3$ python program42.py
Sum of elements in the list =  341
```

```
1   #Sum of elements in the list
2   my_data = [8,23]
3   print "Sum of elements in the list = ", sum(my_data)
```

EXECUTION

```
sh-4.3$ python program42.py
Sum of elements in the list =  31
```

```
1   #Adding two lists
2   list1 = [1,2,3,4,5]
3   list2 = [10,20,30,40,50]
4   sum_list1_list2 = list1 + list2
5   print "Sum of two lists = ", sum(sum_list1_list2)
```

EXECUTION

```
sh-4.3$ python program42.py
Sum of two lists =  165
```

```
1  #Adding two lists
2  list1 = [1,2,3,4,5]
3  list2 = [10,20,30,40,50]
4  list3 = [(x + y) for x, y in zip(list1, list2)]
5  print "Sum of two lists = ", list3
```

EXECUTION

```
sh-4.3$ python program42.py
Sum of two lists =  [11, 22, 33, 44, 55]
```

3.8.6 Simple Calculator Program to Add, Subtract, Multiply and Divide using Functions

PROGRAM 43

```
1  # Simple calculator to add, subtract, multiply and
                                    divide using functions

2
3  # define functions
4  def add(x, y):
5      return x + y
6
7  def subtract(x, y):
8      return x - y
9
10 def multiply(x, y):
11     return x * y
12
13 def divide(x, y):
14     return x / y
15
16 print "Select operation."
```

```
print "1. Add"
print "2. Subtract"
print "3. Multiply"
print "4. Divide"

choice = raw_input("Enter choice(1/2/3/4):")

num1 = input("Enter first number: ")
num2 = input("Enter second number: ")

if choice == '1':
    print num1,"+",num2,"=", add(num1,num2)

elif choice == '2':
    print num1,"-",num2,"=", subtract(num1,num2)

elif choice == '3':
    print num1,"*",num2,"=", multiply(num1,num2)

elif choice == '4':
    print num1,"/",num2,"=", divide(num1,num2)
else:
    print "Invalid input"
```

EXECUTION

```
sh-4.3$ python program43.py
Select operation.
1. Add
2. Subtract
3. Multiply
4. Divide
Enter choice(1/2/3/4):1
Enter first number: 45
Enter second number: 54
45 + 54 = 99
```

```
Select operation.
1. Add
2. Subtract
3. Multiply
4. Divide
Enter choice(1/2/3/4):3
Enter first number: 45
Enter second number: 5
45 * 5 = 225
```

.7 Linear Search is used to Find an Item in an Unordered list

ROGRAM 44

```
1   # Linear search is used to find an item in a unordered list.
2   # To search an item, start at the beginning of the list and
                                        continue searching
3   # until either the end of the list is reached or the item
                                              is found.
4
5   my_data = [89,45,9,21,34] #items in the list, array element
                                        start at location 0
6   num = input("Enter search number = ")
7   for i in range(0,len(my_data)):
8     if num == my_data[i]:          #if item at position i
9   print "Item is location at the position = " , i
```

ECUTION

```
sh-4.3$ python program44.py
Enter search number = 34
Item is location at the position =  4
```

3.8.8 Binary Search

Binary search is a fastest algorithm for searching an element in a sorted array. Running time of binary search is log2 N time, but the searching time of linear search is N/2. Binary search is suitable for large lists and is more efficient sorting algorithm.

The element in the array is described as below and the element to be searched is 45.

Location	0	1	2	3	4	5	6	7	8
Elements	**6**	**12**	**17**	**23**	**38**	**45**	**77**	**84**	**90**

Iteration number	Low	High	Mid
Iteration 1	0	8	4

Search(45)

$$mid = \frac{low + high}{2} = \frac{0 + 8}{2} = 4$$

Location	0	1	2	3	4	5	6	7	8
Elements	**6**	**12**	**17**	**23**	**38**	**45**	**77**	**84**	**90**

Low **Mid** **High**

Iteration number	Low	High	Mid
Iteration 1	0	8	4
Iteration 2	5	8	6

Search(45)

$$mid = \frac{low + high}{2} = \frac{5 + 8}{2} = 6$$

Location	0	1	2	3	4	5	6	7	8
Elements	**6**	**12**	**17**	**23**	**38**	**45**	**77**	**84**	**90**

Low **Mid** **High**

Location	0	1	2	3	4	5	6	7	8
Elements	**6**	**12**	**17**	**23**	**38**	**45**	**77**	**84**	**90**

Low **High** **Mid**

Figure 3.15 Successful Search

```
# Binary search is used to find an item in a ordered list.
def binary_search(item_list,item):
    first = 0
    last = len(item_list)-1
    found = False
    while( first<=last and not found):
        mid = (first + last)//2
        if item_list[mid] == item :
            found = True
            print "Element found at location =", mid
        else:
            if item < item_list[mid]:
                last = mid - 1
            else:
                first = mid + 1
    return found

print "Search the element 45 in the list 1:"
print(binary_search([6,12,17,23,38,45,77,84,90],45))
print "Search the element 55 in the list 2:"
print(binary_search([16,12,17,23,38,45,77,84,90], 5))
```

```
sh-4.3$ python program45.py
Search the element 45 in the list 1:
Element found at location = 5
True
Search the element 55 in the list 2:
False Item is location at the position = 4
```

4.5 ILLUSTRATIVE PROGRAMS

4.5.1 Bubble Sort

Algorithm: Bubble Sort

Step 1: Start from the first element

Step 2: Check two adjacent elements in the list

Step 3: If second element is greater swap the positions

Step 4: Repeat this for entire list

Step 5: Repeat this until no swaps are needed

In bubble sort the list is passed through multiple times. During every pa adjacent items are compared and exchanged if they are out of order. During ea pass the algorithm places the next largest value in the proper place.

Each item "bubbles" up to the appropriate position in the list. Each pass called iteration. In the below figure, for each iteration, the shaded items are t ones that are swapped. If the length of list is n, an element is compared with re of n-1items. The figure 4.6 illustrates the bubble sort for list of nos given iteration 1.

Iteration 1: 51, 1, 34, 12, 8
swapped = True

1	51	34	12	8
1	34	51	12	8
1	34	12	51	8
1	34	12	8	51

Iteration 2: 1, 34, 12, 8, 51
swapped = True

1	12	34	8	51
1	12	8	34	51

Iteration 3: 1, 12, 8, 34, 51
swapped = True

1	8	12	34	51

Iteration 4: 1, 8, 12, 34, 51
swapped = False

Figure 4.6 Bubble Sort of List

thon program for Bubble Sort

PROGRAM 24

```
1. # Bubble Sort
2. Nos_list = [54,26,93,17,77,31,44,55,20]
3. print 'Original List : ', Nos_list
4.
5. #Initially no nos are swapped
6. swapped = False
7.
```

```
8.  while (swapped == False) :
9.          # Pass through entire list
10.         for i in range(0, len(Nos_list)-1):
11.                 # Check the adjacent nos
12.                 if Nos_list[i]>Nos_list[i+1]:
13.                         temp = Nos_list[i]
14.                         Nos_list[i] = Nos_list[i+1]
15.                         Nos_list[i+1] = temp
16.                         swapped = True
17.       #If no swapping has happened break the while loop
18.           if swapped == False:
19.                               break
20.       #Swapping has happened; Continue to chech if further
                                                  sorting is needed
21.           swapped = False
22.
23. print 'Sorted List : ', Nos_list
```

EXECUTION

```
sh-4.3$ python program 24.py
Original List :  [54, 26, 93, 17, 77, 31, 44, 55, 20]
Sorted List :  [17, 20, 26, 31, 44, 54, 55, 77, 93]
```

.5.2 Selection Sort

Algorithm: Selection Sort

Step 1: Assume element in index 0 as MIN

Step 2: Compare MIN will all the other elements in the list

Step 3: If an element lesser than MIN exits; place it in index 0

Step 4: Now assume element in index 1 as MIN; repeat steps 2 & 3

Step 5: Repeat until list is sorted

Selection sort is an improvement of bubble sort. On every pass on the list ly one swap is done. Each time the largest element is pushed to the appropriate lex. After the first iteration, the largest item is placed in the correct index. er the second iteration, the second largest is placed in correct index. This cess continues until the entire list is sorted. For a list with n elements 1)*(n-1) iterations are required to sort.

Figure 4.7 shows the entire sorting process. The list is sorted in descending er in this program. Changing the "<" symbol to ">" will give the list in ascending er. As the lowest element is pushed to end after the iteration, the no of elements en up for sorting in the next iteration is reduced by 1.

In the below figure the smallest element is shaded in black and the elements ded in grey are the ones that are swapped. In an iteration the lowest is ed to its appropriate index and that element is never again considered for parison.

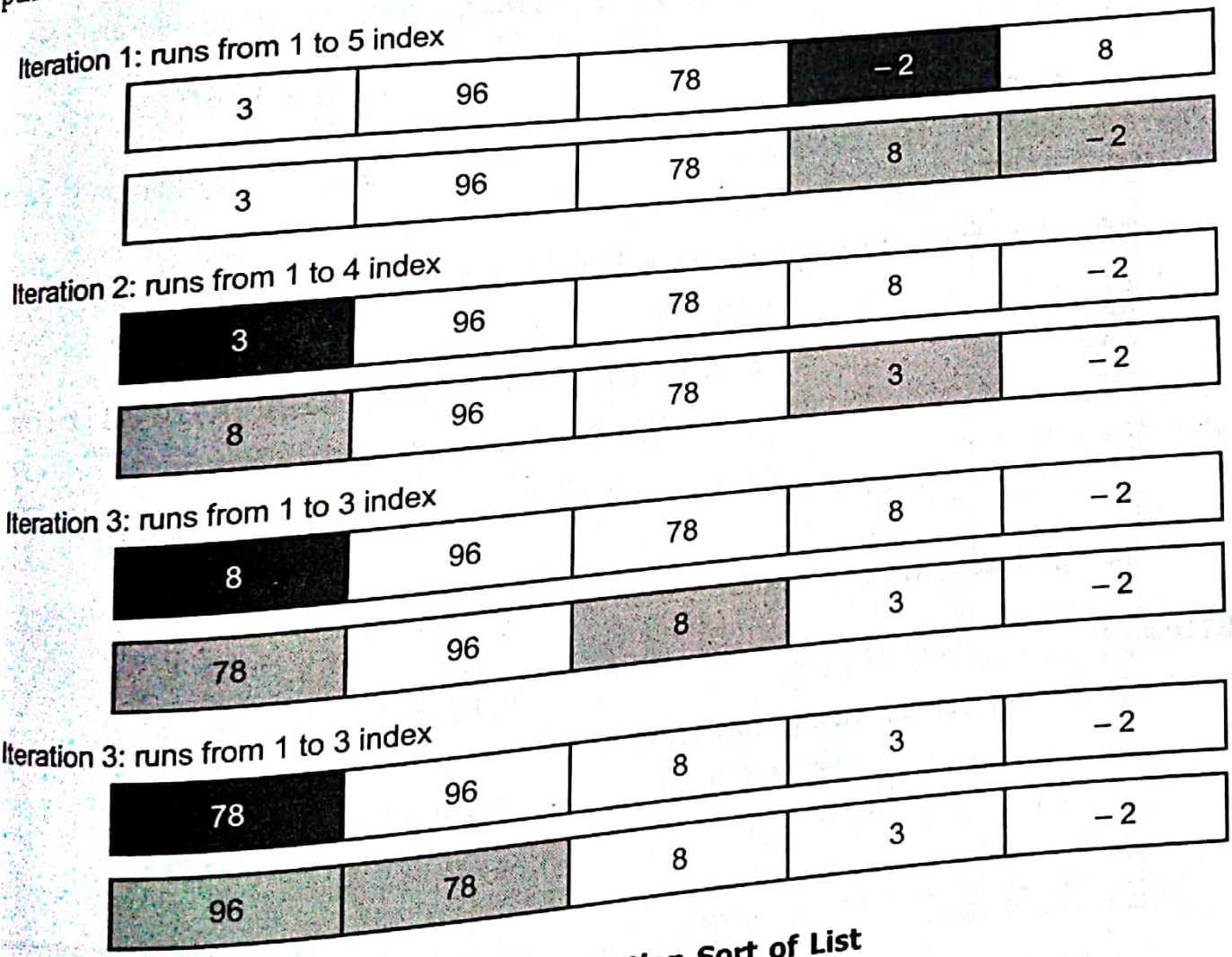


Figure 4.7 Selection Sort of List

```
1. # Selection Sort in Python
2.
3. # Function to swap two nos
4. def swap( Tlist, ind1, ind2 ):
5.                 tmp = Tlist[ind1]
6.                 Tlist[ind1] = Tlist[ind2]
7.               Tlist[ind2] = tmp
8.
9. #List to be sorted
10. Nos_List = [7,2,5,1,29,6,4,19,11]
11. print "Before Sorting : ",Nos_List
12.
13. # For i=8, 7, 6, 5, 4, 3, 2, 1
14. for i in range(len(Nos_List)-1,0,-1):
15.       MinPosition = 0
16.       # Iteration specified in the figure
17.       for k in range( 1 , i+1 ):
18.                 if Nos_List[k] < Nos_List[MinPosition]:
19.                       MinPosition = k
20.                 swap(Nos_List, MinPosition, i)
21.
22. print "After Sorting : ",Nos_List
```

EXECUTION

```
sh-4.3$ python program 25.py
Before Sorting : [7,2,5,1,29,6,4,19,11]
After Sorting : [29, 19, 11, 7, 6, 5, 4, 2, 1]
```

5.3 Insertion Sort

Algorithm: Insertion Sort

Step 1: Compare the first and second element; arrange in order

Step 2: Pick third element; Compare with elements before; arrange in order

Step 3: Pick next element; Compare with all elements before; arrange in order

Step 4: Repeat until till the last element; until the list is sorted

The Insertion Sort algorithm divides the input list into two parts: the sub- t that is sorted, which is maintained in the left end of the list, and the sub-list at is yet to be sorted, which occupy the rest of the list. Initially, the sorted sub- t is empty and the unsorted sub-list is the entire input list. The algorithm rts with the assumption that the sorted sub-list with first item alone is already rted.

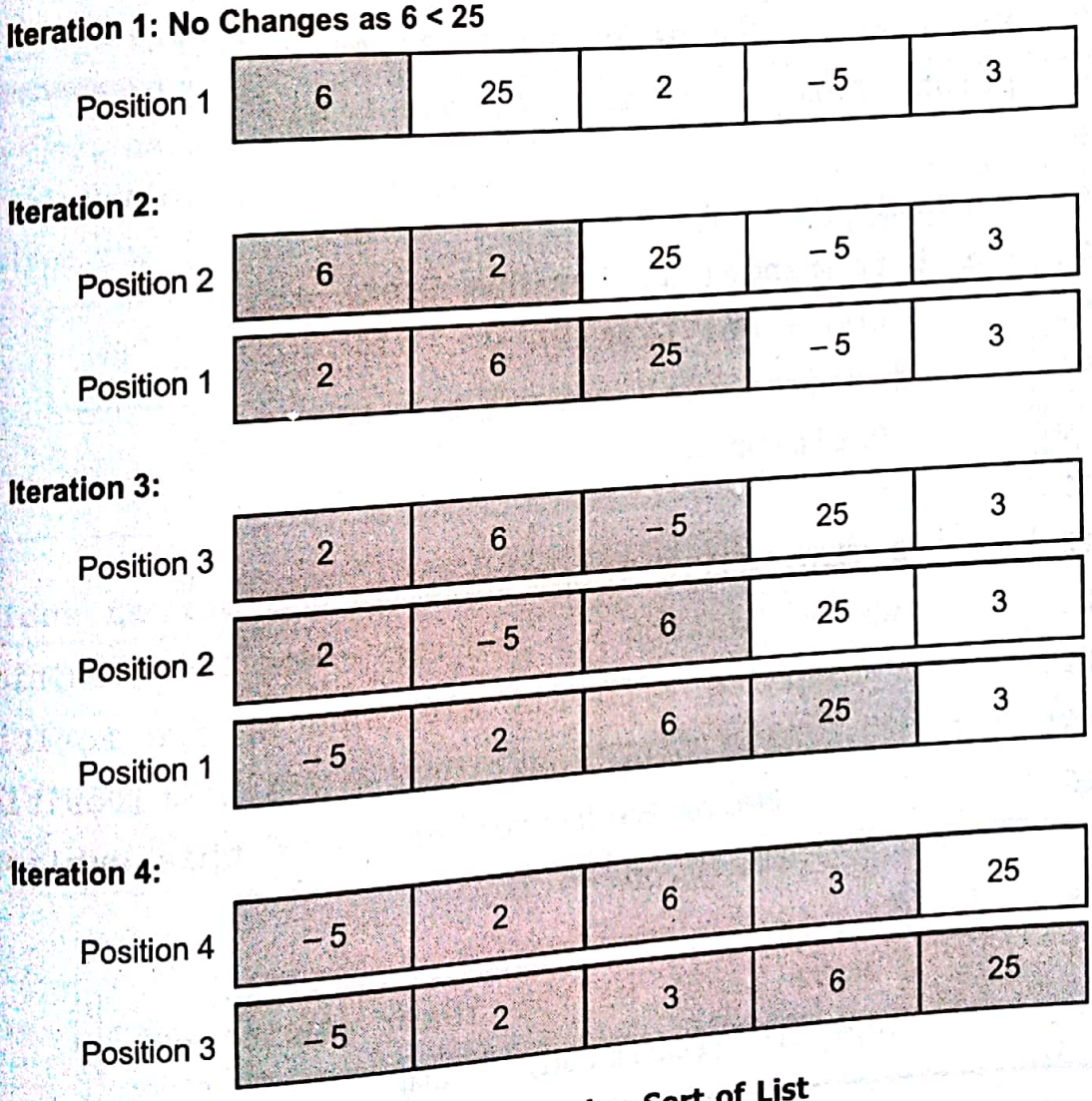


Figure 4.8 Insertion Sort of List

The smallest (or largest, in case of descending order) element in the unsort ub-list is moved to leftmost position in the unsorted sub-list and that becom art of sorted sub-list. The Insertion sort doesn't use the swapping as perform n the previous algorithms but just shifts the elements. Selection sort is efficie n mutable data types like list.

The elements shaded in grey are the elements in sorted sub-list. The re re the elements in unsorted sub-list. After the iteration, the sorted sub-list keep rowing. Each time the least element is passed on to the sorted sub-list and th ub-list is maintained in the correct order by the while loop in line 13.

ython program for Insertion Sort

PROGRAM 26

```
1. # Insertion Sort in Python
2.
3. Nos_List = [54,26,93,17,77,31,44,55,20]
4. print "Before Sorting: ", Nos_List
5.
6. # for i =1 to 9
7. for i in range( 1, len(Nos_List) ):
8.         tmp = Nos_List[i]
9.        # starts with second element
10.        Position = i
11.
12.       # This while loops creates the sorted sublist
13.       while Position > 0 and tmp < Nos_List[Position - 1]:
14.       # Compares with all elements before Position
15.             Nos_List[Position] = Nos_List[Position - 1]
16.             #Move Backwards and sort this sublist
17.             Position -= 1
18.
19.       # Move least to the least position
20.        Nos_List[Position] = tmp
```

```
21.
22. print "After Sorting : ",Nos_List
```

EXECUTION

```
sh-4.3$ python program 26.py
Before Sorting :  [54,26,93,17,77,31,44,55,20]
After Sorting : [17, 20, 26, 31, 44, 54, 55, 77, 93]
```

4.5.4 Merge Sort

Algorithm: Merge Sort

Step 1: If noofItems >= 1 return else continue

Step 2: Divide the list into two lists; firstList and secondList

Step 3: Sort firstList and secondList; by dividing each list further into halves

Step 4: Merge the sorted lists to get the sorted list.

Merge Sort uses divide and conquer strategy. It is a recursive algorithm that continually splits a list into half. A list that has more than one item is split into two halves and recursively the merge sort is invoked on both halves.

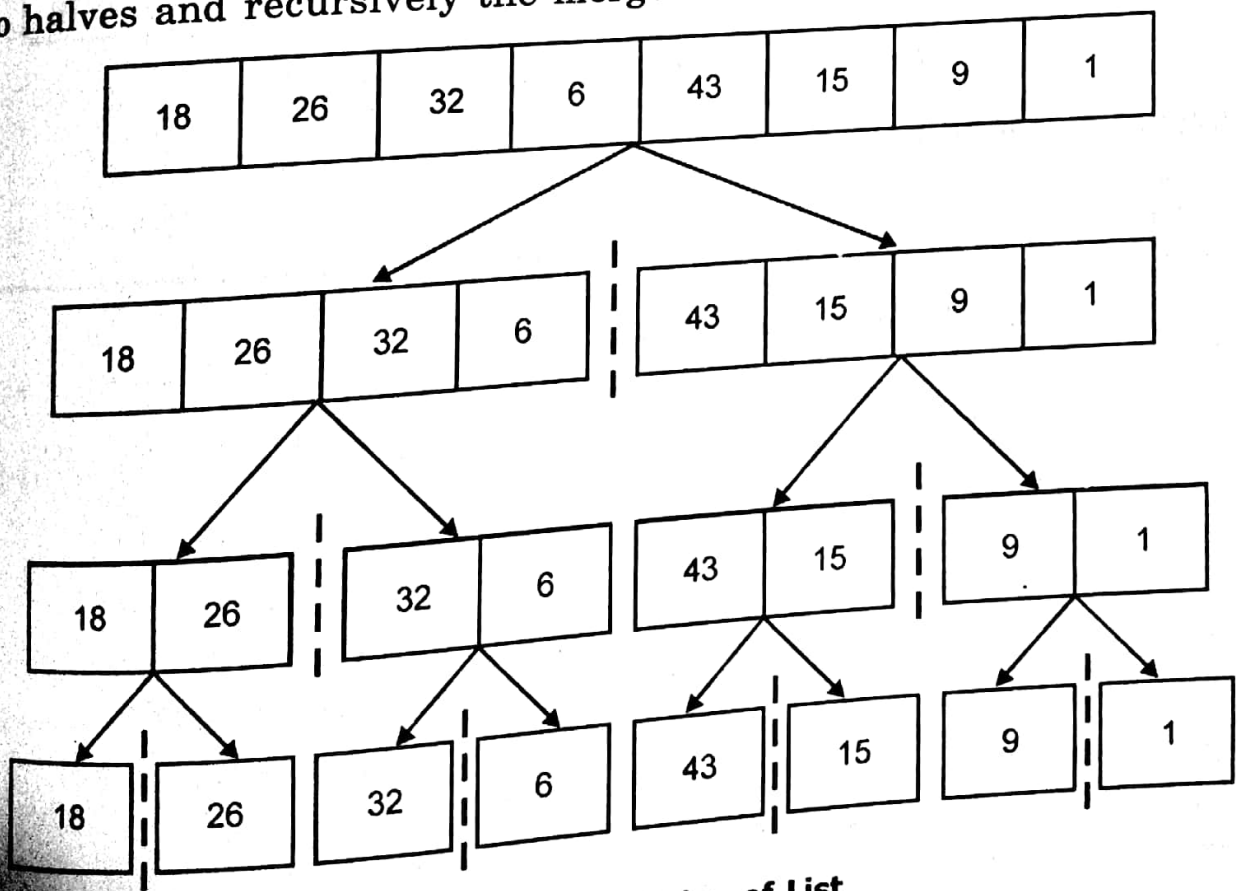


Figure 4.9 Splitting of List

Then the two halves are sorted and merged together. Two sorted lists a
nerged into a single sorted list. The figure 4.9 gives the illustration of how t
sts are split into halves. The figure 4.10 gives the sorted list created by mergi
he sorted lists.

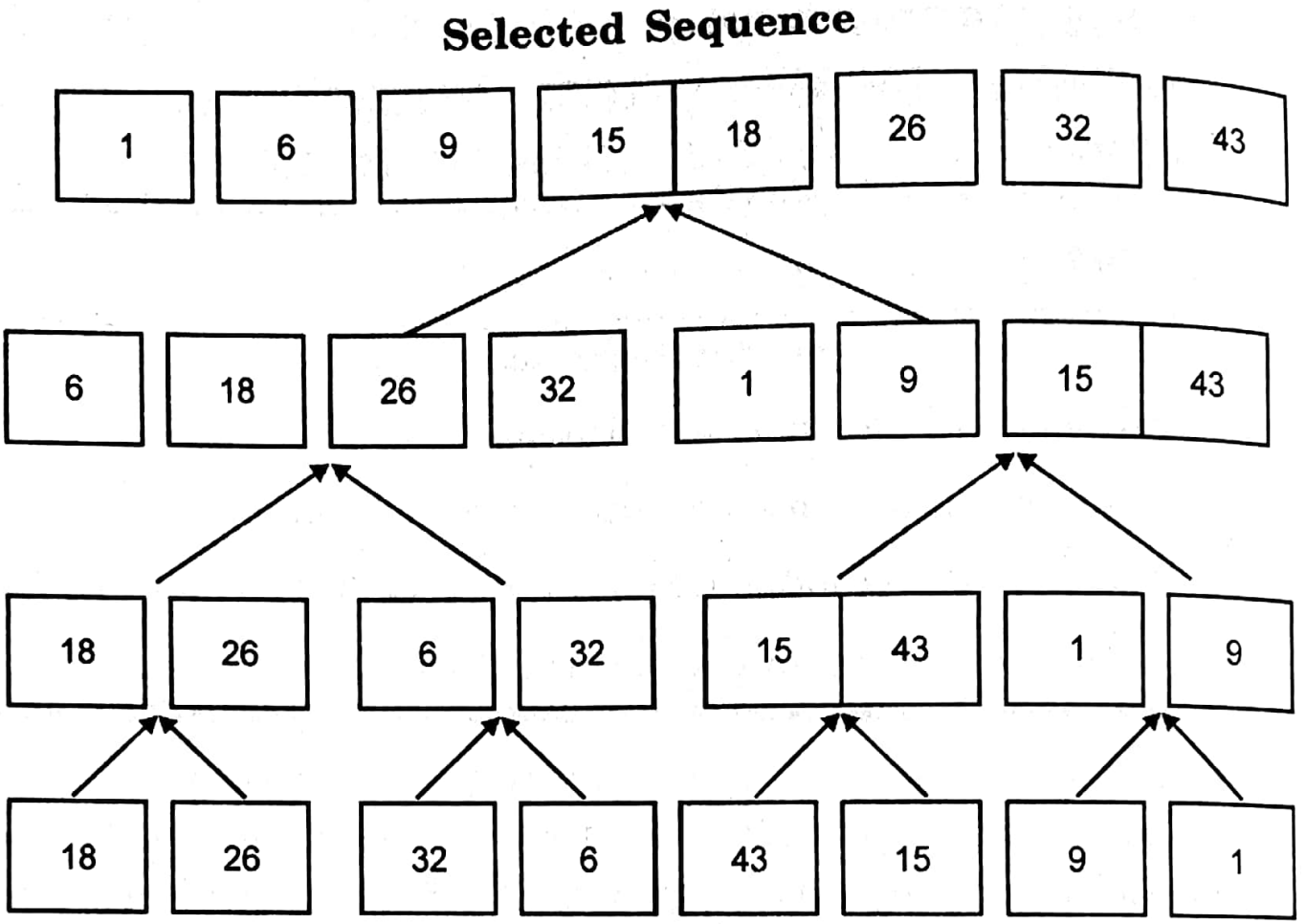


Figure 4.10 Merging of Sorted Lists

ython program for Merge Sort

PROGRAM 27

```
1. # Merge Sort in Python
2.
3. def mergesort( TList, first, last ):
4. # Divide the list into smaller lists
5.         mid = ( first + last ) / 2
6.         if first < last:
7.                 mergesort( TList, first, mid)
8.                 mergesort( TList, mid + 1, last)
9. # merge sorted lists into one
10.        fIndex, lIndex = first, mid + 1
```

```
        finalList = []
#Add to finalList smaller nos
        while fIndex <= mid and lIndex <= last:
                        if TList[fIndex] < TList[lIndex] :
                         finalList.append(TList[fIndex])
                                        fIndex += 1
                        else:
                         finalList.append(TList[lIndex])
                                        lIndex += 1
# End of While; add the remaining sorted elements
        if fIndex <= mid :
        finalList = finalList + TList[fIndex:mid + 1]
        if lIndex <= last:
        finalList= finalList +  TList[lIndex:last + 1]
#Add to TList all sorted nos
        a=0
        while first <= last:
                TList[first] = finalList[a]
                first += 1
                a += 1

#Main Program
Nos_List  =  [54,26,93,17,77,31,44,55,20]
print "Before Sorting : ",Nos_List
mergesort(Nos_List, 0, len( Nos_List ) - 1 )
print "After Sorting : ",Nos_List
```

EXECUTION

```
sh-4.3$ python program 27.py
Before Sorting :  [54,26,93,17,77,31,44,55,20]
After Sorting : [17, 20, 26, 31, 44, 54, 55, 77, 93]
```

4.5.5 Quick Sort

Algorithm: Quick Sort

Step 1: Pick an element and assume it as PIVOT

Step 2: Partition

Arrange elements with values less than the PIVOT before it

Arrange elements with values greater than the PIVOT after it

Step 3: Recursively apply the above steps; for

List of elements with smaller values and greater values separately

The quick sort also uses divide and conquer technique. The algorithm selects the first element as the pivot value.

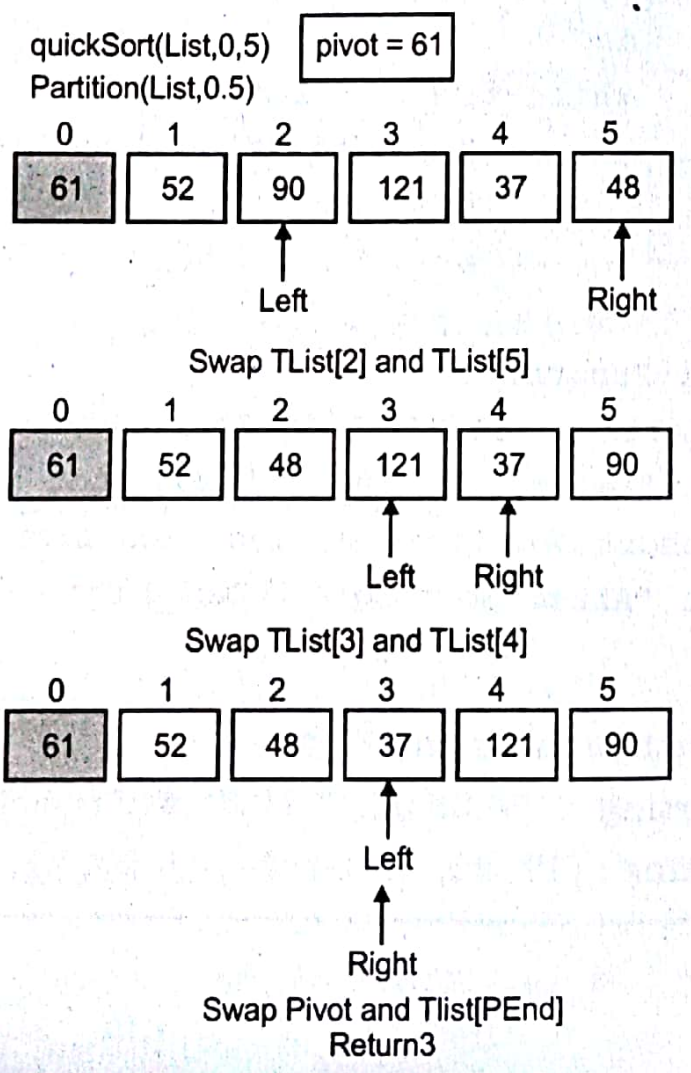


Figure 4.11 Quicksort Iteration 1

The pivot value plays the major role in splitting the list. The index of the ot value in the final sorted list is called the split point. The split point will be d to divide the list for subsequent calls to the quick sort. The bold and shaded ment is the pivot element. The list gets partitioned according to this pivot ment.

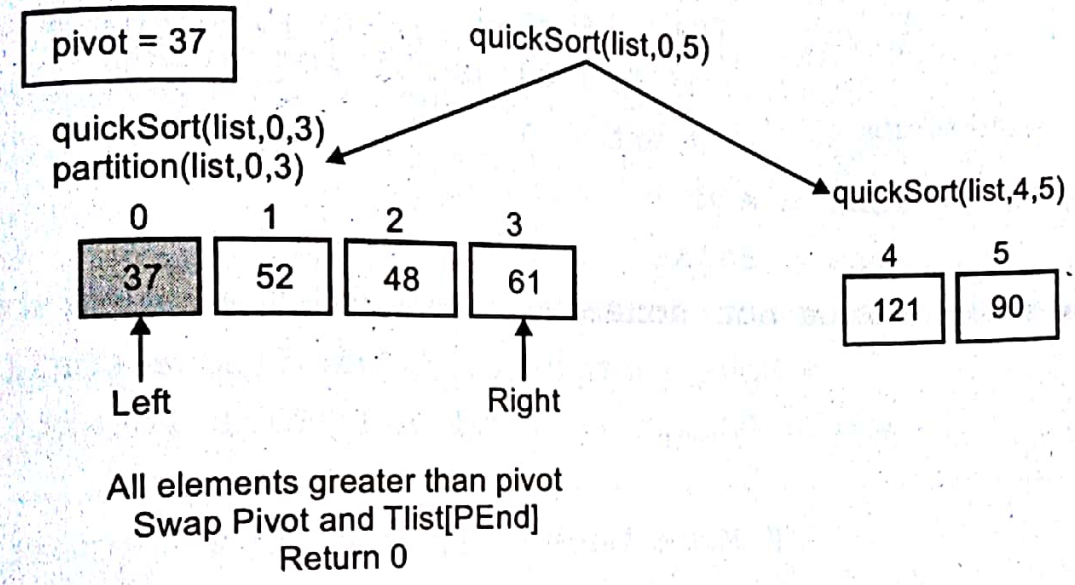


Figure 4.12 Quicksort after 1st Partition

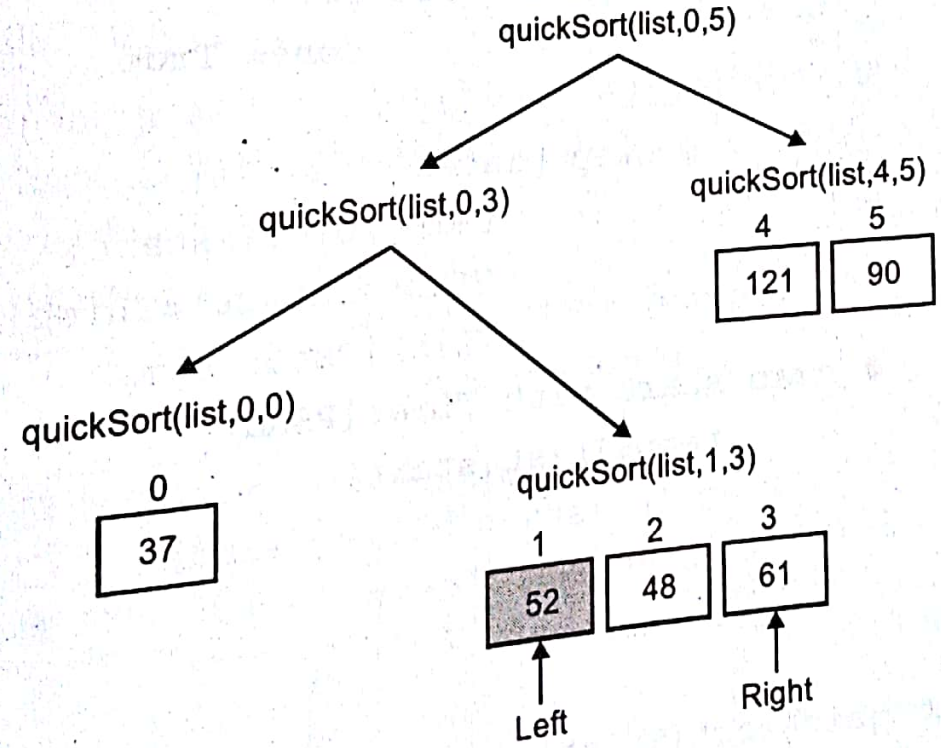


Figure 4.13 Quicksort after 2nd Partition

PROGRAM 28

```
1. # Quick Sort in Python
2. def partition(TList, start, end):
3.         pivot = TList[start]
4.         # Start and End of partition area
5.         PStart = start + 1
6.         PEnd = end
7.         done = False
8.         while not done:
9.             # Move further till you find a smaller element
10.         while PStart <= PEnd and TList[PStart] <= pivot:
11.                         PStart = PStart + 1
12.             # Move back till you find a greater element
13.             while TList[PEnd] >= pivot and PEnd >=PStart:
14.                         PEnd = PEnd - 1
15.                 if PEnd < PStart:
16.                                 done= True
17.                 else:
18.                 # swap places
19.                         temp=TList[PStart]
20.                         TList[PStart]=TList[PEnd]
21.                         TList[PEnd]=temp
22.     # swap start with TList[PEnd]
23.             temp=TList[start]
24.             TList[start]=TList[PEnd]
25.             TList[PEnd]=temp
26.             return PEnd
27.
28. def quicksort(TList, start, end):
29.     if start < end:
30.             # partition the list
31.             partInd = partition(TList, start, end)
32.             # sort both halves
```

```
33.                 quicksort(TList, start, partInd-1)
34.                 quicksort(TList, partInd+1, end)
35.         return TList
36.
37. Nos_List = [7,2,5,1,-29,6,4,-19,11]
38. print "Before Sorting : ",Nos_List
39. sortedList = quicksort(Nos_List,0,len(Nos_List)-1)
40. print "After Sorting : ",sortedList
```

EXECUTION

```
sh-4.3$ python program 28.py
Before Sorting :  [7, 2, 5, 1, -29, 6, 4, -19, 11]
After Sorting :  [-29, -19, 1, 2, 4, 5, 6, 7, 11]
```

4.5.6 Histogram

In normal English, histogram is a diagram consisting of rectangles whose area is proportional to the frequency of a variable and whose width is equal to the class interval.

A histogram is an accurate graphical representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable as show in the below picture. The program 29 tries to create a simple histogram; given the value for each bar.

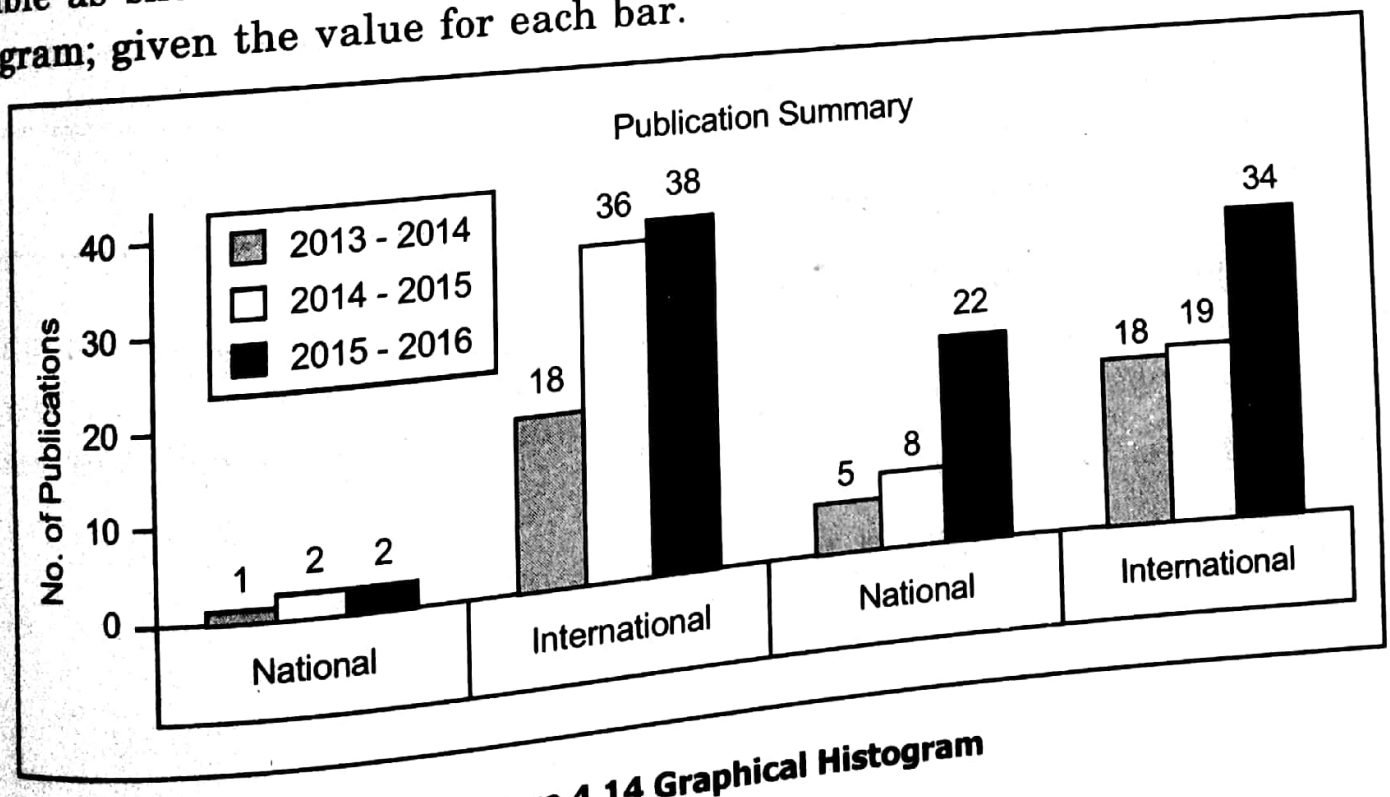


Figure 4.14 Graphical Histogram

PROGRAM 29

```
1. #Example for graphical histogram
2.
3. #Function to create histograms with = symbol
4. def histogram( items ):
5. #For every value print that many = symbol
6. for n in items:
7. output = ''
8. times = n
9. while( times > 0 ):
10. output += '='
11. times = times - 1
12. #Print the value after every output. That's the use of ,
13. print output,
14. print n
15.
16. #Call the function#
17. histogram([20, 33, 50, 15])
```

EXECUTION:

```
sh-4.3$ python main.py
==================== 20
================================= 33
================================================== 50
============== 15
```

In Python, when the data type like list, string or tuple are considered; by histogramwe mean a function that counts the occurrence of element in a. The output of the function is a dictionary with element as key and count as value.

For example; Let L = 'abracadabra'

histogram (L) = {'a': 5, 'b': 2, 'c': 1, 'd': 1, 'r': 2}

PROGRAM 30

```
#Example for occurrencehistogram
#With counter method
from collections import Counter
print  Counter("abracadabra")

#Using List Comprehension
# For list data type
list_nos = [1,2,32,2,4,5,2,4]
print 'Using list and count method: ',
print {x: list_nos.count(x) for x in list_nos}

# For tuple data type
tupleVal = ('hi', 'bye', 'python','book', 'bye','univ',
            'bye','hi')
Dict_histogram= {}
print 'Using dictionary pop method: ',
for x in tupleVal:
    Dict_histogram[x]= Dict_histogram.pop(x,0) + 1
print Dict_histogram

# For mixed list data type
L=[12,'pytho',(12,'check'),'pytho',(12,'check'), 'pytho']
Mix_Dict = {}
print 'Using dictionary setdefault method: ',
for x in L: Mix_Dict[x] = Mix_Dict.setdefault(x,0)+1
print Mix_Dict
```

CUTION

```
sh-4.3$ python program 30.py
Counter({'a': 5, 'r': 2, 'b': 2, 'c': 1, 'd': 1})
Using list and count method:  {32: 1, 1: 1, 2: 3, 4: 2, 5: 1}
Using dictionary pop method: {'python':1, 'bye':3, 'hi': 2, 'univ':1, 'book': 1}
Using dictionary setdefault method:{(12, 'check'): 2, 12: 1, 'pytho': 3}
```

5.9 ILLUSTRATIVE PROGRAMS

5.9.1 File Compare

The following program illustrates the file comparison between two text files. The difference in each line is specified as a list.

PROGRAM 29

```
1.  # Open file for reading in text mode
2.  File1 = open("Version1.txt")
3.  File2 = open("Version2.txt")
4.  Diff_File = open("Difference.txt",'w')
5.
6.  # Read the first line from the files
7.  File1Ln = File1.readline()
8.  File2Ln = File2.readline()
9.
10. # Initialize counter for line number
11. line_no = 1
12.
13. # Check for end of file
14. while File1Ln != '' or File2Ln != '':
15.
```

```
16.      #Remove the whitespaces
17. File1Ln = File1Ln.rstrip()
18.      File2Ln = File2Ln.rstrip()
19.
20.      # If not equal check each word
21.      if File1Ln != File2Ln:
22.
23.          wordsList1 = File1Ln.split()
24.          wordsList2 = File2Ln.split()
25. DiffList = [a for a in wordsList1+wordsList2 if (a
             not in wordsList1) or (a not in wordsList2)]
26. TempStr  = 'Line No:' + str(line_no) +str(DiffList)
27. Diff_File.write(TempStr)
28.      #Read the next line from the file
29. File1Ln = File1.readline()
30.      File2Ln = File2.readline()
31.
32.      #Increment line counter
33. line_no += 1
34. # Close the files
35. File1.close()
36. File2.close()
37. Diff_File.close()
38.
39. Diff_File = open("Difference.txt",'r')
40. print 'The difference between Version1.txt and Version2.txt'
41. print
42. print Diff_File.read()
43. Diff_File.close()
```

Contents if Version1.txt:

```
Hi this is file Compare
The line is matching
Python is too Great
Difference is interesting
End
```

Contents if Version2.txt:

```
Hi this is file copy
The line is matching
Python is Great
Difference is interesting
Bye
```

EXECUTION

```
sh-4.3$ python program 29.py
The difference between Version1.txt and Version2.txt
Line No:1 Difference = ['Compare', 'copy']
Line No:3 Difference = ['too']
Line No:5 Difference = ['End', 'Bye']
```

5.9.2 File Copy

Using Module

Python has standard module; shutil module - to handle operations on file or collection of files. There are several functions for file copy and movement. In this program the function shutil.copyfile(sorc, dest) is used.

shutil.copyfile(sorc, dest)

The contents of the file named sorc are copied to the file named dest. The dest must be the complete target file name. When the dest is already existing the contents are completely removed.

Errors is raised when any of the following criteria is met

1. Both dest and sorc are the same file
2. When the write permission is denied for dest location
3. sorc and dest are not string

PROGRAM 30

```
1.  # File Copy using shutil module
2.  from shutil import copyfile
3.
4.  # Get the file names
5.  sourcefile = input("Enter source file name: ")
```

```
6.  destinationfile=input("Enter destination file name: ")
7.
8.  #copyfile is a function in shutil module
9.  copyfile(sourcefile, destinationfile)
10. print("File copied successfully!")
11. print
12. print("Contents of destination file :")
13. print
14.
15. #Read the file contents of destination
16. FileRead = open(destinationfile, "r")
17. print(FileRead.read())
18. FileRead.close()
```

Contents of 'A.txt':

```
python program
I am using module
Copy is very simple
why try new
old is gold
```

EXECUTION

```
sh-4.3$ python program 30.py
Enter source file name: 'A.txt'
Enter destination file name: 'B.txt'
File copied successfully!
Contents of destination file
python program
I am using module
Copy is very simple
why try new
old is gold
```

Contents of 'B.txt' after execution:

```
python program
I am using module
Copy is very simple
why try new
old is gold
```

Without using module

PROGRAM 31

```
1.  # File Copy without module
2.
3.  # Get the file names
4.  sorc = input("Enter source file name: ")
5.  dest = input("Enter destination file name: ")
6.
7.  sourcefile = open(sorc,'r')
8.  destinationfile = open(dest,'w')
9.
10. #Copy every line from source to destination
11. for line in sourcefile:
12. destinationfile.write(line)
13.
14. #Close files
15. sourcefile.close()
16. destinationfile.close()
17.
18. print("File copied successfully!")
19. print
20. print ("Contents of destination file")
21.
22. #Read the file contents of destination
23. FileRead = open(dest, "r")
24. print(FileRead.read())
25. FileRead.close()
```

Contents of 'A.txt':

```
python program
I am using module
Copy is very simple
why try new
old is gold
```

EXECUTION

```
sh-4.3$ python program 31.py
Enter source file name: 'A.txt'
Enter destination file name: 'B.txt'
File copied successfully!
Contents of destination file
python program
I am using module
Copy is very simple
why try new
old is gold
```

Contents of 'B.txt' after execution:

```
python program
I am using module
Copy is very simple
why try new
old is gold
```

5.9.3 Words Count

Using Module

Python has standard module; the collections module is used for the containe datatypes like list, dict, set and tuple.

collections.Counter(str)

The Counter function in the collections module returns a dictionary objec and takes up string as argument. The dictionary has elements that are keys an values pairs in which the keys are words and value is the count of words in th string.

PROGRAM 32

```
1.  #Words Count using module collections
2.  from collections import Counter
3.  FileName = input("Enter the file name : ")
4.  CntFile = open(FileName, 'r')
5.
6.  print("Number of words in the file :")
7.  print(Counter(CntFile.read().split()))
8.  CntFile.close()
```

Contents of 'A.txt' after execution:

```
python program
I am using module that is very simple
module is very simple module
Old module is very simple
```

XECUTION

```
sh-4.3$ python program 32.py
Enter the file name : 'A.txt'
Number of words in the file :  Counter({'module': 4, 'is': 3, 'very': 3,
'simple': 3, 'Old': 1, 'that': 1, 'python': 1, 'am': 1, 'I': 1, 'program': 1,
'using': 1})
```

thout using Module

ROGRAM 33

```
1.  #Word Count without module
2.  FileName = 'Source.txt'
3.
4.  Count_Dict = {}
5.  CntFile = open(FileName,'r')
6.
7.  #For each line in the file count
8.  for line in CntFile:
9.      #Split line into words
```

```
10. word_list = line.split()
11.       #For every word in the list
12.    for word in word_list:
13.   # First time adding into dictionary
14. if word not in Count_Dict:
15. Count_Dict[word] = 1
16. else:
17.   # update dictionary with incremented count.
18. Count_Dict[word] = Count_Dict[word] + 1
19.
20. print ("Count of each word in the file :")
21. print('{:15}{:3}'.format('Word','Frequency'))
22. print('-' * 25)
23.
24. # printing the words and its occurrence.
25. for   (word,count) in Count_Dict.items():
26. print('{:15}{:3}'.format(word,count))
```

EXECUTION

```
sh-4.3$ python program 33.py
Count of each word in the file :
Word            Frequency
-------------------------
Old             1
python          1
is              4
module          4
very            4
without         1
using           1
simple          4
```