2105 – DMI COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GE 3151 - PROBLEM SOLVING AND PYTHON PROGRAMMING

UNIT I ALGORITHMIC PROBLEM SOLVING

PART A

1. What is an algorithm? (*University question*)

Algorithm is an ordered sequence of finite, well defined, unambiguous instructions for completing a task. It is an English-like representation of the logic which is used to solve the problem. It is a step- by-step procedure for solving a task or a problem. The steps must be ordered, unambiguous and finite in number.

2. Write an algorithm to find minimum of 3 numbers in a list. (*University question*)

ALGORITHM : Find Minimum of 3 numbers in a list

- Step 1: Start
- Step 2: Read the three numbers A, B, C
- Step 3: Compare A and B.
  If A is minimum, go to step 4 else go to step 5 Step 4: Compare A and C.
  If A is minimum, output "A is minimum" else output "C is minimum". Go to step 6.
- Step 5: Compare B and C.
  If B is minimum, output "B is minimum" else output "C is minimum".
- Step 6: Stop

3. List the building blocks of an algorithm.

- Statements
- Sequence
- Selection or Conditional
- Repetition or Control flow
- Functions

4. **Define statements. List its type**

Statements are instructions in Python designed as components for algorithmic problem solving. For example. An input statement collects a specific value from the user for a variable within the program. An output statement writes a message or the value of a program variable to the user's screen.

5. Write the pseudo code to calculate the sum and product of two numbers and display.

```
BEGIN
INITIALIZE variables sum, product, number1, number2
READ number1, number2 sum = number1+ number2
PRINT “The sum is “, sum
COMPUTE product = number1 * number2
PRINT “The Product is “, product
END
```

6. What is a function?

Functions are "self-contained" modules of code that accomplish a specific task. Functions usually "take in" data, process it, and "return" a result. Once a function is written, it can be used over and over again. Functions can be "called" from the inside of other functions.

7. Write the pseudo code to calculate the sum and product displaying the answer on the monitor screen.

```
BEGIN
INITIALIZE variables sum, product, number1, number2 of type real
READ number1, number2 sum = number1 +number2
PRINT “The sum is “, sum
COMPUTE product = number1 * number2
PRINT “The Product is “, product
END program
```

8. **Give the rules for writing Pseudo codes.**

- Write one statement per line.
- Capitalize initial keywords.
- Indent to show hierarchy.
- End multiline structure.
- Keep statements to be language independent

.**9. Give the difference between flowchart and pseudo code.**

- Flowchart is a graphical representation of the algorithm.
- Pseudo code is a readable, formally English like language representation.

10. Define a flowchart.

A flowchart is a diagrammatic representation of the logic for solving a task. A flowchart is drawn using boxes of different shapes with lines connecting them to show the flow of control. The purpose of drawing a flowchart is to make the logic of the program clearer in a visual form.

11. Give an example of Iteration. a = 0

```
for i from 1 to 3    // loop three times
a = a + i            // add the current value of i to a
print a              // the number 6 is printed (0 + 1; 1 + 2; 3 + 3)
```

12. Write down the rules for preparing a flowchart.

- A flowchart should have a start and end,
- The direction of flow in a flowchart must be from top to bottom and left to right
- The relevant symbols must be used while drawing a flowchart.

13. Mention the characteristics of an algorithm.

- Algorithm should be precise and unambiguous.
- Instruction in an algorithm should not be repeated infinitely.
- Ensure that the algorithm will ultimately terminate.
- Algorithm should be written in sequence.
- Algorithm should be written in normal English.
- Desired result should be obtained only after the algorithm terminates.

14. Define the two modes in Python.

Python has two basic modes: script and interactive.

The normal mode is the mode where the scripted and finished .py files are run in the Python interpreter. Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory

15. Give the various data types in Python

Python has five standard data types

- Numbers
- String
- List
- Tuple
- Dictionary

16. List out the simple steps to develop an algorithm.

Algorithm development process consists of five major steps.

- Step 1: Obtain a description of the problem.
- Step 2: Analyze the problem.
- Step 3: Develop a high-level algorithm.
- Step 4: Refine the algorithm by adding more detail.
- Step 5: Review the algorithm.

17. Give the differences between recursion and iteration

Recursion	Iteration
Function calls itself until the base condition is reached.	Repetition of process until the condition fails.
Only base condition (terminating condition) is specified.	It involves four steps: initialization, condition, execution and updation.
It keeps our code short and simple.	Iterative approach makes our code longer.
It is slower than iteration due to overhead of maintaining stack.	Iteration is faster.
It takes more memory than iteration due to overhead of maintaining stack.	Iteration takes less memory.

18. What are advantages and disadvantages of recursion?

Advantages	Disadvantages
Recursive functions make the code look clean and elegant.	Sometimes the logic behind recursion is hard to follow through.
A complex task can be broken down into simpler sub-problems using recursion.	Recursive calls are expensive (inefficient) as they take up a lot of memory and time.
Sequence generation is easier with recursion than using some nested iteration.	Recursive functions are hard to debug.

19. Define control flow statement.

A program's control flow is the order in which the program's code executes. The control flow of a Python program is regulated by conditional statements, loops, and function calls.

20. Write an algorithm to accept two number. compute the sum and print the result. (*University question*)

Step 1: Start

Step 2: READ the value of two numbers

Step 3:Compute sum of two numbers

Step 4 :Print the sum

Step 5:Stop

21. Write an algorithm to find the minimum number in a given list of numbers. (*University question)*

- Step 1: Start
- Step 2:Read the total number of element in the list as N
- Step3:Read first element as E
- Step 4:MIN=E
- Step 5:Set i=2
- Step6:IF i>n goto Step 11 ELSE goto Step 7
- Step 7:Read i th element as E
- Step 8:IF E<MIN then set MIN=e
- Step 9:i=i+1
- Step 10:goto step 6
- Step 11:print MIN
- Step12:Stop

22. Outline the logic to swap the contents of two identifiers without using the third variable *(University question)*

- Step1: Start
- Step 2:Read A,B
- Step 3 :A=A+B
- Step 4:B=A-B
- Step5:A=A-B
- Step 6:Print A ,B
- Step 7:Stop

<u>PART – B</u>

1. Algorithm with an example.

a. Characteristics
b. Phases
c. Qualities
d. Representation

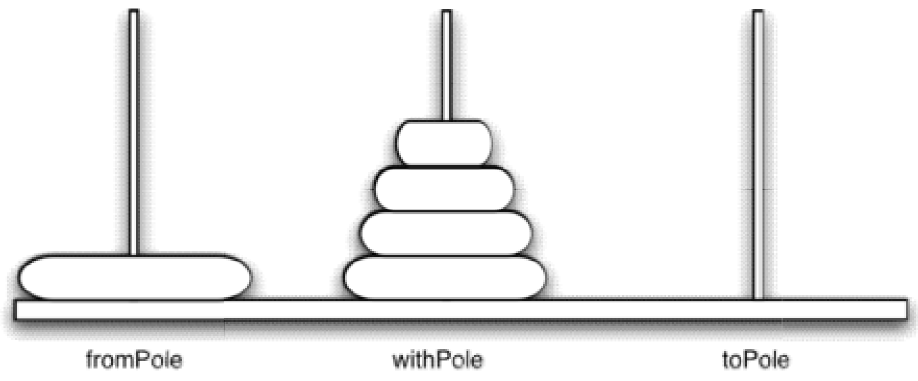
2. **Building blocks of an algorithm with an example.**
   a. Instructions / Statement
      i. Simple Statement
      ii. Compound Statement
   b. State
   c. Control Flow
      i. Sequence Control Structure
      ii. Selection Control Structure
      iii. Case Structure
   d. Functions

3. **Towers of Hanoi problem with an example.**

This should be done with two important constraints.

- We can only move one disk at a time.
- We cannot place a larger disk on top of a smaller one.

Figure 1 shows an example of a configuration of disks in the middle of a move from the first pole to the third pole.

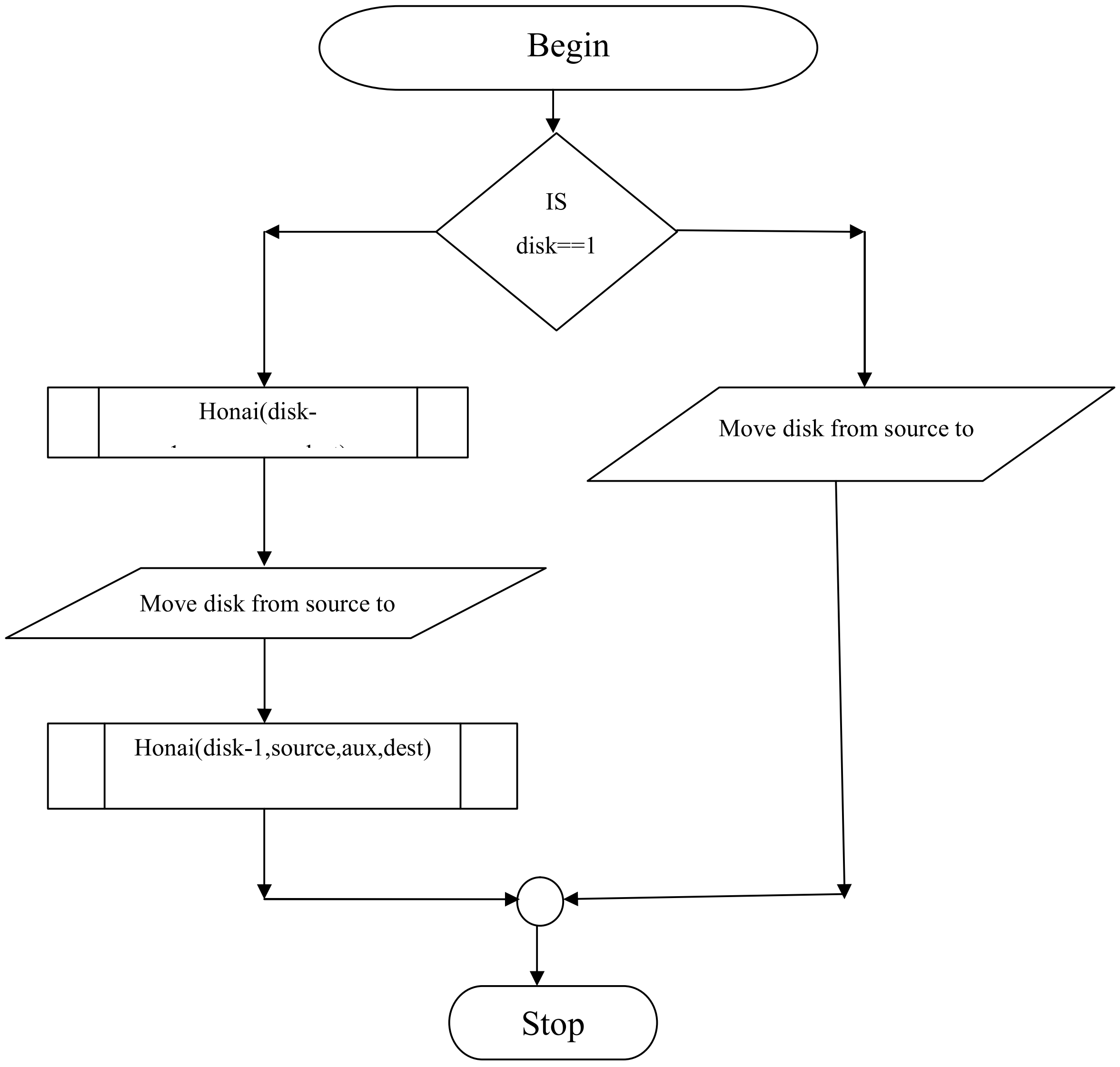


<u>A</u> ***<u>recursive</u>*** ***<u>Step</u>*** ***<u>based</u>***

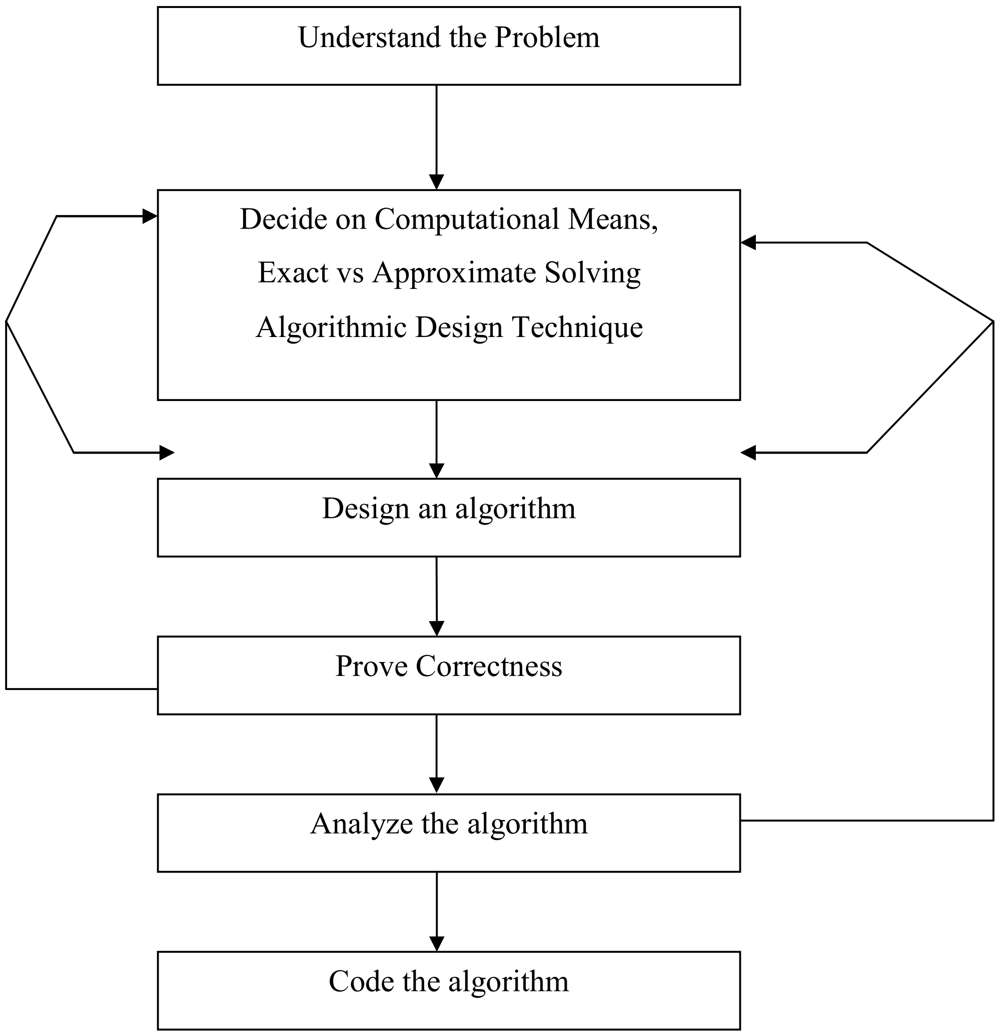
algorithm for Tower of Hanoi

```
Step 1: BEGIN Hanoi(disk, source, dest, aux)
Step 2: IF disk == 1 THEN go to step 3 ELSE go to step 4
Step 3: move disk from source to dest AND go to step 8
Step 4: CALL Hanoi(disk - 1, source, aux, dest)
Step 5: move disk from source to dest
Step 6: CALL Hanoi(disk - 1, aux, dest, source)
Step 7: END Hanoi
```

FLOW CHART



4. Arithmetic problem solving with a neat sketch.



5. Explain in detail about the Notation of an algorithm with an example

a. Pseudo code
   i. Guidelines
   ii. Representation
   iii. Advantages and Disadvantages
b. Flow Chart
   i. Guidelines
   ii. Representation
   iii. Advantages and Disadvantages
c. Programming Language
   i. Low Level Language / Machine Language

ii. Intermediate level Language / Assembly level Language

iii. High Level Language / Natural Language

6. **Strategies of an algorithm with an example.**

   a. Iteration

   b. Recursion

UNIT II DATA, EXPRESSIONS, STATEMENTS

PART – A

1. What is a value? What are the different types of values? (University question)

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

2. Define a variable and write down the rules for naming a variable.

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.

3. Define keyword and enumerate some of the keywords in Python. (University question)

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.

4. Define statement and what are its types?

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

5. What do you meant by an assignment statement?

An assignment statement creates new variables and gives them values:

Eg 1: Message = “hello”

Eg 2: n = 17

6. What is tuple? (*University question*)

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)

7. What is an expression?

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

Examples:

Y=x + 17

8. What do you mean by an operand and an operator? Illustrate your answer with relevant example.

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.

9. What is the order in which operations are evaluated? Give the order of precedence.

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.

10. Illustrate the use of * and + operators in string with example.

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

Example:

```
>>> 'Hello*3'
```

Output: HelloHelloHello

```
>>>'Hello+World'
```

Output: HelloWorld

11. What is the symbol for comment? Give an example.

# is the symbol for comments in python.

Example:

```
# compute the percentage of the hour that has elapsed
```

12. What is function call?

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.

13. Identify the parts of a function in the given example.

```
>>>     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.

14. What is a local variable?

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

15. What is the output of the following?

a. float(32)

b. 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.

16. What do you mean by flow of execution?

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.

17. Write down the output for the following program.

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

Output:

```
throatwarbler
```

18. Give the syntax of function definition.

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

19. Explain the concept of floor division.

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

20. What is type coercion? Give example.

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 / 60.0
0.983333333333
```

21. Write a math function to perform √2 / 2.

```
>>>     math.sqrt(2) / 2.0
        0.70710678118
```

22. What is meant by traceback?

A list of the functions that tells us what program file the error occurred in, and what line, and what functions were executing at the time. It also shows the line of code that caused the error.

23. Write a program to accept two numbers multiply them and print them. (*<u>University question</u>*)

```
A=10
B=2
multiplication=A*B
print ("Multiplication of two numbers :" ,multiplication)
```

<u>PART – B</u>

1. **Modes of Python with an example.**
    a. Python Interpreter
    b. Python Interactive mode
    c. In Script mode
2. **Explain Values and Data types with an example.**
    a. Integer
    b. Float
    c. Boolean
    d. String
    e. List
    f. Tuple
    g. Dictionary
3. **Operators**

a. Arithmetic Operator
b. Comparison Operator (or) Relational Operator
c. Assignment Operator
d. Logical Operator
e. Bitwise Operator
f. Membership Operator
g. Identity Operator

4. Precedence Of Operators

S. No	Operators	Description
1.	()	Parentheses
2.	**	Exponent
3.	+x, -x, ~x	Unary plus, Unary minus, Bitwise NOT
4.	*, /, //, %	Multiplication, Division, Floor division, Modulus
5.	+, -	Addition, Subtraction
6.	<<, >>	Bitwise shift operators
7.	&	Bitwise AND
8.	^	Bitwise XOR
9.	|	Bitwise OR
10.	==, !=, >, >=, <, <=,	is, is not, in, not in Comparison, Identity, Membership operators
11.	not	Logical NOT
12.	and	Logical AND
13.	or	Logical OR

5. Associativity Of Python Operators

Precedence	Operator	Description
1	**, ()	Exponent, Inside Parenthesis
2	/, *, %, //	Division, Multiplication, Modulo, Floor
3	+, -	Addition, Subtraction

Example:

2 ** 3 ** 2 is equivalent to 2 ** (3 ** 2).

6. Variables, Statement, Expression, Keywords

UNIT III CONTROL FLOW, FUNCTIONS

GE3151 - Problem Solving and Python Programming

PART – A

1.Define Boolean expression with example.

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

Eg :

```
>>>     5 == 6
>>>     False
```

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

2. What are the different types of operators?

- 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)

3. Explain modulus operator with example.

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.

4. Explain relational operators.

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
```

5. Explain Logical operators(University question)

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.

6. What is conditional execution?

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 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.

7. What is alternative execution?

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'
```

8. What are chained conditionals?

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.

9. Explain while loop with example. 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:

1. Evaluate the condition, yielding True or False.
2. If the condition is false, exit the while statement and continue execution at the next statement.
3. If the condition is true, execute the body and then go back to step 1

10. Explain ‘for loop’ with example.

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
```

11. What is a break statement?

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!'
```

12. What is a continue statement?

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
```

13.Compare return value and composition.

Return Value:

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
```

Composition:

Calling one function from another is called composition.

Eg:

```
def circle_area(xc, yc, xp, yp):
        radius = distance(xc, yc, xp, yp) result = area(radius)
        return result
```

14.What is recursion?

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)
```

15. Explain global and local scope.

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")
```

16.Compare string and string slices.

A string is a sequence of character.

Eg: fruit = 'banana'

16. 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
```

17. Define string immutability.

Python strings are immutable. 'a' is not a string. It is a variable with string value. We can't mutate the string but can change what value of the variable to a new string.

18. Mention a few string functions.

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

19. What are string methods?

A method is similar to a function—it takes arguments and returns a value—but the syntax is different. For example, the method upper takes a string and returns a new string with all uppercase letters. Instead of the function syntax upper(word), it uses the method syntax word.upper().

```
>>> word = 'banana'
>>>     new_word = word.upper()
>>>     print new_word
BANANA
```

20. Explain about string module.

The string module contains number of useful constants and classes, as well as some deprecated legacy functions that are also available as methods on strings.

Eg: import string

string.upper(text) → converts to upper case

string.lower(text) → converts to lower case

21. What is the purpose of pass statement?

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.

PART – B

1. **Control Structures**
    a. Decision Making (or) Conditionals (or) Branching
        i. if statement
        ii. if-else statement
        iii. if-elif-else statement
        iv. Nested Conditional
    b. Iteration (or) Looping Statement
        i. while loop
        ii. for loop
        iii. Nested loop
    c. Types of Unconditional looping Statement
        i. break statement
        ii. continue statement
        iii. pass statement
2. **Functions / fruitful functions**
    a. Definition

b. Advantages of Function

c. Components of Functions

   i. Function Definition

   ii. Function Calling

   iii. Return Keyword

   iv. Parameters and Arguments

d. Types of function

   i. Built-in Function

   ii. User Defined Function

e. Flow of Execution

3. **Recursion function**

Syntax:

```
def function(parameter):
                    #Body of function
```

(Example Towers of Hanoi)

4. Return values / parameters,

a. Functions with Return values

b. Functions with multiple Return values

5. Local and global scope

Local scope Example:

```
def   outer_function():
    a='I am in india'
    def  inner_function():
        a="I am in TamilNadu"
        print("a=",a)
    inner_function()
    print("a=",a)
a="I am in world"
outer_function()
print("a=",a)
```

Global scope Example:

```
def   outer_function():
    global a
    a="I am in india"
    def  inner_function():
        a=" I am in TamilNadu"
        print("a=",a)
    inner_function()
    print("a=",a)
a="I am in world "
outer_function()
print("a=",a)
```

6. Function composition

A Composition is calling one function from another function definition.

Example:

Write a python program to add three numbers by using function:

```
def addition(x,y,z):                    #Function 1
    add=x+y+z
    return add
def get():                              #Function 2
    a=int(input("Enter first number:"))
    b=int(input("Enter second number:"))
    c=int(input("Enter third number:"))
    print("The addition is:",addition(a,b,c))    #Composition function calling
get()                                            #function calling
```

7. Strings

a. Methods or functions

b. String Slicing

UNIT IV LISTS, TUPLES, DICTIONARIES

PART – A

1. What is a list?

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.

2. 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]
```

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

```
>>>     list = [’a’, ’b’, ’c’, ’d’, ’e’, ’f’]
>>>     list[1:3] [’b’, ’c’]
>>>     list[:4] [’a’, ’b’, ’c’, ’d’]
>>>     list[3:] [’d’, ’e’, ’f’]
```

4. Mention any 5 list methods.

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

5. State the difference between lists and dictionary.

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 hashable whereas list can store a sequence of objects in a certain order.

6. What is List mutability in Python? Give an example.

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]
```

7. What is aliasing in list? Give an example.

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
```

8. Define cloning in list.

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”.

9. Explain List parameters with an example.

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)
```

10. Write a program in Python to delete first element from a list.

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

Here's how deleteHead is used:

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

11. Write a program in Python returns a list that contains all but the first element of the given list.

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

Here's how tail is used:

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

12. What is the benefit of using tuple assignment in Python?

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
```

13.Define key-value pairs.

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.

14. Define dictionary with an example.

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'
```

15. How to return tuples as values?

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)
```

16. List two dictionary operations.

- Del -removes key-value pairs from a dictionary
- Len - returns the number of key-value pairs

17.Define dictionary methods with an example.

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']
```

18.Define List Comprehension.

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.

19.Write a Python program to swap two variables.

```
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))

20.Write the syntax for list comprehension.

The list comprehension starts with a '[' and ']', to help us remember that the result is going to be a list. The basic syntax is [expression for item in list if conditional].

PART – B

1. **List methods or List Functions**

- A list is an ordered set of values, where each value is identified by an index
- Consider the values of list a and list b be
  >>>a=[‘apple’,’mango’,’lime’]
  >>>b=[‘grape’]
- append() -listname.append() The method append() will add the item to the end of a list a.append(‘orange’)
- insert() -listname.insert(index,item) This method inserts an item at a particular place and two arguments (index,item) a.insert(1,’banana’)
- extend() listname.extend(item1, item2) This method is used to combine two list with the items in the argument. a.extend(‘grape’) (or) a.extend(b)
- remove() listname.remove(item) This method will remove an item in the list. a.remove(’apple’)
- remove() listname.remove(item) This method will remove an item in the list. a.remove(’apple’)
- index() listname.index(item) This method will return index value of list and takes index value as argument. a.index(‘lime’)
- copy() dest_list=listname.cop y() This method is used to copy a list to another list. c=a.copy()
- reverse() listname.reverse() This method is used to reverse the items in a list. a.reverse()
- count() listname.count(item) This method is used to count the duplicate items in the list which takes the item as arguments. a.count(‘lime’)
- sort() listname.sort() This method is used to arrange the list from ascending to descending alphabetically. a.sort() >>>a=[‘apple’, ‘lime’, ‘mango’]

2. **List Slicing:**

   A subsequence of a sequence is called a slice and the operation that extracts a subsequence is called slicing.

For slicing we use square brackets [ ].

Two integer values splitted by (:).

Syntax: Ex: >>>a=[‘a’,’b’,’c’,’d’,’e’]

List a= ‘a’ ‘b’ ‘c’ ‘d’ ‘e’

Index from Left 0 1 2 3 4

Index from Right -5 -4 -3 -2 -1

List_Name[Starting_Value : Ending_Value]

3. **List comprehension:**

- List comprehension is an elegant way to define and create list in Python.
- These lists have often the qualities of sets.
- It consists of brackets containing an expression followed by a for clause, then zero or more for or if clauses.
- **Ex-2:**

```
>>>squares = []
>>>for x in range(10):
squares.append(x**2)
print (squares)
>>>squares = [x**2 for x in range(10)] # List comprehensions to get the same result:
>>>print (squares)
Output:
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

4. **Dictionary Methods / functions:**

- len() len(dictonary) Gives the total length of the dictionary. len(d) ◊ 4
- keys() dictionary.keys() Return the dictionary's keys. >>>d.keys() ['a', 'c', 'b', 'd']
- values() dictionary.values() Return the dictionary's Values >>>d.values() [1, 3, 2, 4]
- items() dictionary.items() Return the dictionary's Keys and Values >>>d.items() [('a', 1), ('c', 3), ('b', 2), ('d', 4)]
- key in dict key in dict Returns True if the Key is in Dictionary, else returns false. >>> 'a' in d True
- key not in dict key not in dict Returns True if the Key is not in Dictionary, else returns false. >>> 'e' not in d True

- has_key (key) dictionary.has_key(key) Checks whether the dictionary has the specified key. >>>d.has_key('a') True
- get() dictionary.get(key) Get the value and Return the value of key. >>>d.get('b') 2
- update() Dest_dictionary.update(Source_dict) Update the dictionary with the key from existing keys. >>>d.update(d1) >>> print(d) {'a': 1, 'c': 3, 'b': 2, 'e': 5, 'd': 4, 'f': 6}
- cmp() cmp(dictionary1, dictionary2) Compares elements of both dictionary. Returns 0 if the elements are same, Else returns 1. >>>cmp(d,d1) 1
- copy() new_dict = original_dict.copy() Return a copy of the dictionary. >>> d2=d1.copy() >>> print(d2) {'e': 5, 'f': 6}
- pop() dictionary.pop('key') Remove the item with key and return its value >>>d.pop('f') 6
- popitems() dictionary.popitem() Remove and return an item with its key and value. >>>d.popitem() ('a', 1)
- clear() dictionary.clear() Remove all items form the dictionary. d.clear()

5. **Tuples methods / functions:**
   - Tuples are sequence of values much like the list.
   - The values stored in the tuple can be of any type and they are indexed by integers.
   - The main difference between list and tuple is Tuple is immutable. Tuple is represented using '( )'

   Syntax:

   tuple_name=(items)

   Ex:

   >>> tuple1=('1','2','3','5')

   >>>tuple2=('a','b','c')

   >>>tuple3='3','apple','100'

Tuples methods:

- Python Tuple Methods.
- Python Tuple – max() Method.
- Python Tuple – min() Method.
- Python Tuple – index() Method.
- Python Tuple – len() Method.
- Python Tuple count() Method.
- Python | Remove empty tuples from a list.

- Python | Reversing a Tuple.

6. **Tuples are immutable**

The values of tuple cannot be changed.

```
>>> tuple1=('1','2','3','5')
tuple1[1]='4'         #It Shows Error
```

Tuples can be immutable but if you want to add an item we can add it by t1=('a','b')

t1=('A',)+t1[1:] ◊ t1=('A','b)

The disadvantage in this method is we can only add the items from the beginning.

```
>>>tuple1 = (1, 2, 33, 44, 6)

>>>tuple1[4] = 10

TypeError: 'tuple'object does not support item assignment
```

We can call tuples immutable tuples as we cannot make changes in a tuple once we create it.

UNIT V FILES, MODULES, PACKAGES

PART – A

1. What is a text file?

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

2. Write a python program that writes "Hello world" into a file.

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

3. Write a python program that counts the number of words in a file.

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

4. What are the two arguments taken by the open() function?

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

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

5. What is a file object?

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.

6. What information is displayed if we print a file object in the given program?

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

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

7. What is an exception?

Whenever a runtime error occurs, it creates an exception. The program stops execution and prints an error message. For example, dividing by zero creates an exception:

```
print 55/0
ZeroDivisionError: integer division or modulo
```

8. What are the error messages that are displayed for the following exceptions?

a. Accessing a non-existent list item → IndexError: list index out of range

b. Accessing a key that isn't in the dictionary → KeyError: what

c. Trying to open a non-existent file → IOError: [Errno 2] No such file or directory: 'filename'

9. What are the two parts in an error message?

The error message has two parts: the type of error before the colon, and specific about the error after the colon.

10. How do you handle the exception inside a program when you try to open a non-existent file?

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

11. How does try and execute work?

The try statement executes the statements in the first block. If no exception occurs, then except statement is ignored. If an exception of type IOError occurs, it executes the statements in the except branch and then continues.

12. What is the function of raise statement? What are its two arguments?

The raise statement is used to raise an exception when the program detects an error. It takes two arguments: the exception type and specific information about the error.

13. What is a pickle?

Pickling saves an object to a file for later retrieval. The pickle module helps to translate almost any type of object to a string suitable for storage in a database and then translate the strings back in to objects.

14. What are the two methods used in pickling?

The two methods used in pickling are pickle.dump() and pickle.load(). To store a data structure, dump method is used and to load the data structures that are dumped , load method is used.

15. What is the use of the format operator?

The format operator % takes a format string and a tuple of expressions and yields a string that includes the expressions, formatted according to the format string.

16. What are modules?

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>

17. What is a package?

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

Syntax: from <mypackage> import <modulename>

18. What is the special file that each package in Python must contain?

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

19. How do you delete a file in Python?

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)

20. How do you use command line arguments to give input to the program?

Python sys module provides access to any command-line arguments via sys.argv. sys.argv is the list of command-line arguments. len(sys.argv) is the number of command-line arguments.

PART – B

1. Explain various types of file operation in detail in python

i. OpentheFile
ii. Read orWrite datainthefile
iii. Closethe File

i .Open theFile

Python has a built-in function open () to open a file. This function returns a file object and has two arguments which are **file name & opening mode**. The opening modes are reading, writing or appending.

Syntax:

file_object=open("filename","mode")

File opening modes = "r", "W" "x" , "a", "b", "t", "+"

ii) read( ) Method

This method reads a string from an open file.

Syntax:

```
file_object.read()
```

write()Method

This method writes a string from an open file.

Syntax:

```
file_object.write("String")
```

ii. ClosetheFile

To close a file object we will use close() method. The file gets closed and cannot be used again for reading and writing.

Syntax:

file_object.close()

2**. Discuss about format operator in file processing**

Theargumentofwritehastobeastring,soifwewanttoputothervaluesinafile,

wehaveto convert them to strings. The easiest waytodo that is with str:

```
>>>f=open('stringsample.txt','w')
>>>f.write(5)              #TypeError
>>>f.write(str(5))
```

An alternative is to use the **format operator**, %. The first operand is the **format string**, which contains one or more **format sequences**, which specify how these cond operand is formatted. The result is a string.

Some of the format strings /conversions are. d, I, o, u, x, X, e, E, f, F

3. What are exceptions? Explain the methods to handle with an example.

Exception

An Exception is an event which occurs during the execution of a program that disrupts the normal flow of the program's instructions. If a python script encounters a situation it cannot cope up, it raises an exception.

i) try block

In the try block a programmer will write the suspicious code that may raise an exception. One can defend their program from a run time error by placing the codes inside the try block.

ii) except statement

Except statement should be followed by the try block. A single try block can have multiple except statement. The except statement which handles the exception. Multiple except statements require multiple exception names to handle the exception separately.

iii) else block

If there is no exception in the program the else block will get executed.

iv) finallyblock:

A finally block will always execute whether an exception happens or not the block will always execute.

Syntax-

```
try:
   #The operations here
except Exception 1:
   #HandleException1
except Exception 2:
   #HandleException2
else:
   #Ifno Exception, it will execute
finally:
   #Always Execute
```

4. Explain Command line argument in python with example

It is possible to pass some values from the command line to python programs when they are executed. These values are called command line arguments and it can be used to control program from outside instead of hard coding.

The command line arguments are handled using sys module. We can access command-line arguments via the **sys.argv**

This serves two purposes −

- sys.argv is the list of command-line arguments.
- len(sys.argv) is the number of command-line arguments.Heresys.argv[0] is the program name.

Consider the following script command.py

```
>>>import sys
>>>program_name=sys.argv[0]
>>>arguments=sys.argv[1:]
>>>count=len(arguments)
>>>print(program_name)
>>>print(arguments)
>>>print(count)
```

5. Explain about File manipulation in Python

File manipulation means change or access the content in the file.

1. File Positions

There are two methods to access the positions of the file.

- tell()– syntax - file_object.tell()
- seek() –syntax-file_object.seek(offset,from)

2. Renaming and Delete a File

Two file processing operations are there,

They are

- rename( ) method - syntax-os.rename(current_filename,new_filename)
- remove( ) method - syntax-os.remove(filename)

6. Directories in Python

All files are contained within various directories. The os module has several methods to create, remove and change

directories.mos.remove(filename)

kdir(), chdir()