Functions

Very Short Answer Type Questions (1 marks each)

Question 1:
What is sys.argv?
Answer:
A list containing the program name and all the command-line arguments to a program.

Question 2:
What can you do with the eval function?
Answer:
Eval(s) evaluates an expression contained in a string s and returns the resulting object.

Question 3:
What does sys.argv[1:] mean?
Answer:
A list of the command-line arguments.

Question 4:
What type of errors does the exception type NameError correspond to?
Answer:
A variable that is not initialized (defined), perhaps a variable/function that is misspelled.

Question 5:
Why we use ceil(x) function?
Answer:
It returns the ceiling of x as a float, the smallest integer value greater than or equal to x.

Question 6:
Why we use floor(x) function?
Answer:
It returns the floor of x as a float, the largest integer value less than or equal to x.

Question 7:
Explain fabs(x) in python?
Answer:
It returns the absolute value of x.

Question 8:
What value will be return by exp(x) ?
Answer:
Returns e**x.

Question 9:
What value will be return by log10(x) ?
Answer:
Returns the base-10 logarithm of x. This is usually more accurate than log(x,10).

Question 10:
What value will be return by sqrt(x)?
Answer:
Returns the square root of x.

Question 11:
What value will be return by cos(x)?
Answer:
Returns the cosine of x radians.

Question 12:
Write the output of sin(O)?
Answer:
Returns the zero.

Question 13:
Define tan(x) function in python.
Answer:
Tan(x) function return the tangent of x radians.

Question 14:
Why we use degrees(x) ?
Answer:
It converts angle x from radians to degrees.

Question 15:
What is the advantage of radians(x) ?
Answer:
It converts angle x from degrees to radians.

Question 16:
What value will be return by random() ?
Answer:
Return the next random floating point number in the range (0.0,1.0).

Question 17:
“A module cannot contain a main program”. Is this statement true?
Answer:
False

Question 18:
What gets printed? Assuming python version 2.x print type(l/2)
Answer:
<type’int’>

Question 19:
What is the output of the following code? print type([l,2])
Answer:
< type’list’>

Question 20:
What is the output of the following code? def f(): pass print type(f())
Answer:
<type’NoneType’>

Question 21:
What should the below code print?
print type(1J)
Answer:
< type’complex’>

Question 22:
What will be the output of the following code

```
class c (object):
….def_init_(self):
self.x = 1
C = C()
print C.X
print C.X
```

```
print C.X
print C.X
```

Answer:

All the outputs will be 1, since the value of the objects attribute (X) is never changed.

```
1
1
1
1
```

X is now a part of the public members of the class C. Thus it can be accessed directly.

Question 23:

What gets printed with the following code ?

```
d= lambda p: p * 2
t = lambda p: p * 3
x = 2
x = d(x)
x = t(x)
x = d(x)
print x
```

Answer:

24

Question 24:

What gets printed with the following code ?

```
x = 4.5
y = 2
print x//y
```

Answer:

2.0

Question 25:
What gets printed with the following code ?
print “hello” ‘world’
Answer:
On one line the text: helloworld

Question 26:
What gets printed by the code snippet below?
import math
print math.floor(5.5)
Answer:
5.0

Question 27:
What gets printed by the code snippet below?
x = “foo ”
y = 2
print x + y
Answer:
An exception is thrown.

Question 28:
What does the following code do?
def a(b, c, d): pass
Answer:
It defines a function, which does nothing.

Question 29:
What do you mean by function?
Answer:
A function is a block of organized, reusable code that is used to perform a single, related action.

Question 30:
Do Python supports built-in functions?
Answer:
Yes, Python gives you many built-in functions like print() etc.

Question 31:
Can First statement of a function be optional?
Answer:
Yes, The first statement of a function can be an optional statement.

Question 32:
Why we use return [expression] in Python?
Answer:
The statement return [expression] exits a function.

Question 33:
Write syntax for function that takes a string as input parameter and prints it on standard screen.
Answer:
def printme (str):
"This prints a passed string into this function"
print str
return

Question 34:
Why do we define a function?
Answer:

- Decomposing complex problems into simpler pieces.
- Reducing duplication of code.

Question 35:
When we can execute a function?
Answer:
Once the basic structure of a function is finalized, we can execute it

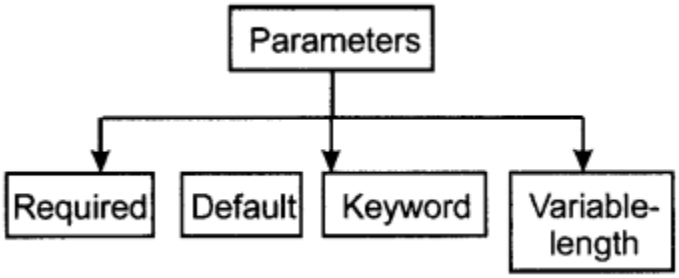
Question 36:
Are all parameters (arguments) in the Python language passed by reference.
Answer:
Yes,All parameters (arguments) in the Python language are passed by reference.

Question 37:
How do we define a function ?
Answer:
Here are simple rules to define a function in Python.

- Function blocks begin with the keyword def followed by the function name and Parentheses ()
- Any input parameters or arguments should be placed within these parentheses.
- The first statement of a function can be an optional statement – the documentation string of the function or docstaing.

● he code block within every function start with color(:) and is indented.
● The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return none.

Question 38:
What are the various types of para-meters ?
Answer:



Question 39:
What are Required arguments?
Answer:
Required arguments are the arguments passed to a function in correct positional order.

Question 40:
Explain default argument.
Answer:
A default argument is an argument that assumes a default value, if a value is not provided in the function call.

Question 41:
Write the syntax of lambda functions?
Answer:
The syntax of lambda functions contains only a single statement, which is as follows:
lambda[arg1 [,arg2, argn]] :expression

Question 42:
What is the use of lambda keyword in Python?
Answer:
You can use the lambda keyword to create small anonymous functions.

Question 43:
Explain return keyword.
Answer:
The statement return [expression] exits a function, optionally passing back an expression to the caller.

Question 44:
Define scope of a variable.
Answer:
The scope of a variable determines the portion of the program where you can access a particular identify.

Question 45:
What are two basic scopes of variables in Python?
Answer:
Global variables and Local variables.

Question 46:
What does the following code do? def simpleFunction():
"This is a cool simple function that returns 1"
return 1
print simpleFunction._doc_[10:14]
Answer:
cool

Question 47:
Which variables have local scope?
Answer:
Variables that are defined inside a function body have a local scope.

Question 48:
Which variables have global scope?
Answer:
Variables that are defined outside a function body have a global scope.

Short Answer Type Questions (2 marks each)

Question 1:
What is a Python module ? What is its significance ? .
Answer:
A 'module' is a chunk of python code that exists in its own (.pu) file and is intended to be used by python code outside iteself. Modules allow one to fundle together code in a form in which it can easily be used later.
The modules can be 'imported' in other programs so the function and other definitions in imported modules becomes availabe to code that imports them.

Question 2:
"Python has certain functions that you can readily use without having to write any special code." What type of functions are these ?
Answer:
The pre-defined functions that are always available for use are known as python's built-in functions. For example :
len (), type (), int (), raw-input () etc.

Question 3:
What is the utility of built-in function help () ?
Answer:
Python's built in function help ( ) is very useful. When it is provided with a program-name or a module-name or a function-name as an argument, it displays the documentation of the argument as help. It also displays the docstrings within its passed-argument's definition. For example :
help (math)
will display the documentation related to module math.

Question 4:
What is the utility of Python standard library's math module and random module ?
Answer:
Math module is used for math related functions that work with all number types except for complex numbers, radom module is used for different random number generator functions.

Question 5:
What is raw_input?
Answer:
It is a function which takes a string (e.g., a question) as argument, writes the string to the terminal window, halts the program and lets the user write in text on the keyboard, and then the text is returned to the calling code as a string object.

Question 6:
What value will be return by log(x[, base])?
Answer:
With one argument, return the natural logarithm of x (to base e).
With two arguments, return the logarithm of x to the given base, calculated as log(x) log(base)

Question 7:
Define pow(x, y) function in Python ?
Answer:
Return x raised to the power y.. In particular, pow(1.0, x) and pow(x, 0.0) always return 1.0,

even when x is a zero or a NaN. If both x and y are finite, x is negative, and y is not an integer then pow(x, y) is undefined, and raises ValueError.

Question 8:

Determine what is printed by the following program listinput.py,

```
list1 = eval(raw_input('Give a list:'))
list2 = eval(raw_input('Give another list:'))
print list1 + list2
```

When the program is running as follows:

```
Unix/DOS > python listinput.py
Give a list: '[1,2]'
Give another list: "[3,4]"
```

Answer:

[1,2] [3,4]

Question 9:

What will be the output of this program?

```
a = 20.3
b = 'Python'
c = '"Python"'
d = 21
print isinstance(a, float),
print isinstance(b, (str, int)),
print isinstance(c, (str, str)),
print isinstance(d, (float, str))
```

Answer:

True, True, True, False.

Question 10:

Write any Python expression which is a valid input argument to the following program?

```
input = eval(raw_input('Value:'))
print 'You typed', value
```

Answer:

'one'

Question 11:

Given the program

```
from scitools.StringFunction import
StringFunction
import sys
formula = sys.argv[1]
f = StringFunction(formula)
```

print f(2)
Will this input work?
Unix/DOS > python function.py 't**2'
Answer:
No, This input will not work

Question 12:
What will happen when you run the following program:
import sys
try:
t = sys.arg[1]
a = sys.arg[2]
except:
print 'You must provide TWO
command-line arguments!!!'
sys.exit(1)
s = 0.5*a*t**2
print s
Answer:
The program will always respond with, "You must provide TWO command-line arguments!!!", regardless of what you write on the command line

Question 13:
How can you make a module 'helloworld' out of these two functions?
def hello():
print 'Hello,',

def world(): print 'World!'
Answer:
Put the functions in a file with name helloworld.
py

Question 14:
Write a program to read the user name using raw_input() and display back on the screen using print()
Answer:
#!/usr/bin/python
name=raw_input('Enter your name :')
print ("Hi %s, Let us be friends!" % name);

Question 15:
What does the len() function do?

Answer:
It gets the length of the string that you pass to it then returns it as a number. Play with it.

Question 16:
How do you make a higher order function in Python ?
Answer:

● A high order function accepts one or more functions as input and returns a new function. Sometimes it is required to use function as data.
● To make high order function, one need the import functools module.
● The functools.partial() function is used often for high order function.

Question 17:
What value will be return by “choice(seq)” ?
Answer:
A random item from a list, tuple, or string.

Question 18:
What value will be return by “random()” ?
Answer:
A random float r, such that 0 is less than or equal to r and r is less than 1.

Question 19:
Give an example of “choice()”
Answer:

The following example shows the usage of choice( ) method.
# !/usr/bin/py thon
import random
print “choice([l, 2, 3, 5, 9]): “,
random.choice([l,2,3,5,9])
print “choice(‘A String*): “,
random.choice(‘A String’)

This will produce the following result choice

([l, 2, 3,5,9]): 2
choice(‘A String’) :n

Question 20:
Give an example of “random()”

Answer:
The following example shows the usage of random() method.

```
# !/usr/bin/python
import random
# First random number
print “random(): ” ,
random.random() # Second random number print “random(): “,
random.random()
```

This will produce the following results

```
random(): 0.281954791393
random(): 0.309090465205
```

Question 21:
Describe trignometric functions ?
Answer:
Python includes following functions that perform trignometric calculations.

Function	**Description**
acos(x)	Return the arc cosine of x, in radians.
asin(x)	Return the arc sine of x, in radians.
atan(x)	Return the arc tangent of x, in radians.
atan2(y, x)	Return atan(y/x), in radians.
cos(x)	Return the cosine of x radians.
hypot(x, y)	Return the Euclidean norm, sqrt(x*x + y^y).
sin(x)	
tan(x)	
degrees(x)	Converts angle x from radians to degrees.
radians(x)	Converts angle x from degrees to radians.

Question 22:
Describe random functions ?
Answer:
Random numbers are used for fames, simulations, testing, security, and privacy applicatons.
Python includes following functions that are commonly used.

Function	Description
choice(seq)	A random item from a list, tuple, or string.

randrange ([start,] stop [,Step])	A randomly selected element from range(start, stop, step)
random()	A random float, such that 0 is less than or equal to r and r is less than 1
seed([x])	Sets the integer starting value used in generating random numbers. Call this function before calling any other random module function. Returns None.
shuffle(Ist)	Randomizes the items of a list in plae. Returns None.
Uniform(x, y)	A random float r, such that x is less than or equal to r and r is less than y

Question 23:
Write a program to swap two numbers.
Answer:
```
a = 5
b = 9
def swap(c,d):
return d,c
swap(a,b)
```

Question 24:
What are the rules to define a function in Python?
Answer:
You can define functions to provide the required functionality. Here are simple rules to define a function in Python:

(1) Function blocks begin with the keyword def followed by the function name and parentheses (( ))
(2) Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
(3) The first statement of a function can be an optional statement-the documentation string of the function or docstring.
(4) The code block within every function starts with a colon (:) and is indented.
(5) The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

Question 25:
Give an example for defining a function.
Answer:
```
def functionname(parameters):
“function_docstring” function_suite
return[expression]
```

Question 26:
Write a program to illustrate function call.
Answer:

```
#!/usr/bin/python # Function definition is here
def printme(str): “This prints a passed string into this function”
print tr;
return; # Now you can call printme function printme(“Mohd saif naqvi”);
printme(“Chairman of Waltons Technology”)
```

It produces following result:

```
Mohd saif naqvi
Chairman of Waltons Technology
```

Question 27:

Write a program to illustrate “Pass by refe-rence vs value”

Answer:

```
#!/usr/bin/python # Function definition is here
def changeme(mylist): “This changes a passed list into this function”
mylist.append([l,2,3,4]);
print “Values inside the function: “, mylist
return # Now you can call changeme
function mylist=[10,20,30];
changeme(mylist);
print “Values outside the function: “, mylist
```

It produce following result:

```
Values inside the function: [10, 20,30, [1,2,3,4]]
Values outside the function: [10,20, 30, [1,2,3,4]]
```

Question 28:

Write a program to illustrate “Required arguments”.

Answer:

```
#!/usr/bin/python
# Function definition is here
def printme (str): “This prints a passed string into this function”
print str;
return;
# Now you can call printme function printme();
```

It produces following result:

```
Traceback (most recent call last):
File “test.py”, line 11, in <module>
printme()
TypeError: printme() takes exactly 1 argument (0 given)
```

Question 29:

Write a program to illustrate “Keyword arguments”.

Answer:

```
#!/usr/bin/python
# Function definition is here
def printme(str):"Thisprints a passed string into this function"
print str;
return;
# Now you can call printme function
printme(str="My string");
```

When the above code is executed, it produces following result:

My string

Question 30:

Write a program to illustrate "default argument".

Answer:

```
#!/usr/bin/python
# Function definition is here
def printinfo(name,age=35):
"This prints a passed info into this function"
print "Name: ",
name; print "Age ",
age return;
# Now you can call printinfo function
printinfo(age=50,name="miki");
printinfo(name="miki");
```

When the above code is executed, it produces following result:

Name: miki
Age 50
Name:miki
Age 35

Question 31:

Write a program to illustrate "variablelength arguments".

Answer:

```
#!/usr/bin/python
# Function definition is here
```

```
def printinfo(argl,*vartuple):
"This prints a variable passed arguments"
print "Output is:"
print argl for var in vartuple :
print var return;
# Now you can call printinfo function
printinfo(10); printinfo(70,60,50);
```

When the above code is executed, it produces following result:

```
Output is:
10
Output is:
70
60
50
```

Question 32:
Explain Global vs. Local variables.
Answer:
Variables that are defined inside a function body have a local scope, and those defined outside have a global scope.
This means that local variables can be accessed only inside the function in which they are declared whereas global variables can be accessed throughout the program body by all functions. When you call a function, the variables declared inside it are brought into scope.

Question 33:
Explain anonymous functions.
Answer:
You can use the lambda keyword to create small anonymous functions. These functions are called anonymous because they are not declared in the standard manner by using the def keyword, lambda functions have their own local namespace and cannot access variables other than those in their parameter list and those in the global namespace.

Question 34:
Explain Scope of Variables.
Answer:
All variables in a program may not be accessible at all locations in that program. This depends on where you have declared a variable.
The scope of a variable determines the portion of the program where you can access a particular identifier. There are two basic scopes of variables in Python :

(1) Global variables
(2) Local variables

Question 35:
What do you mean by module ?
Answer:
A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitarily named attributes that you can bind and reference.
Simply, a module is a file consisting of Python code. A module can define functions, classes, and variables. A module can also include runnable code.

Question 36:
What do you mean by globals() and locals () Functions.
Answer:
The globals( ) and locals( ) functions can be used to return-the names in the global and local namespaces depending on the location from where they are called.
If locals() is called from within a function, it will return all the names that can be accessed locally from that function.
If globals() is called from within a function, it will return all the names that can be accessed globally from that function .The return type of both these functions is dictionary. Therefore, names can be extracted using the keys() function

Question 37:
Explain reload)) Function.
Answer:
When the module is imported into a script, the code in the top-level portion of a module is executed only once.Therefore, if you want to re-execute the top -level code in a module, you can use the reload() function.
The reload() function imports a previously imported module again. The syntax of the reload)) function is this :
reload (module_name)

Question 38:
What are docstrings ? How are they useful ?
Answer:
A docstring is just a regular python triple- quoted string that is the first thing in a function body or a module or a class. When executing a functionbody the docstring does not do anything like comments, but Python stores it as part of the function documen-tation. This documentation can later be displayed using help)) function. So even though docstrings appear like comments but these are different from comments.

Question 39:
Find the error
def minus (total, decrement)
output = total – decrement
return output
Answer:
The function's header has colon missing at the end.
So add colon (:) at end of header line.
Thus, the correct code is

def minus (total, decrement)
output = total – decrement
return output

Question 40:
What would be the output produced by the following code:
import math
import random
print math=ceil (random.random)))
Answer:
The output would be :
1.0
random.random() would generate a number in the range [0.0, 1.0] but math.ceil ( ) will return ceiling number for this range, which is 1.0 for all the numbers in this range.
Thus the output produced will always be 1.0.

Question 41:
Find the error(s) in the following code and correct them:

1. def describe intelligent life form ():
2. height = raw_input ("Enter the height")
3. raw_input ("Is it correct ?")
4. weight = raw_input ("Enter the weight")
5. favorite-game = raw_input ("Enter favorite game")
6. print "your height", height, 'and weight', weight
7. print "and your favorite game is", favorite- game, '.'

Answer:
Here, function name contains spaces. But function name should be a valid Python identifier with no spaces in between. And here, no variable is defined to obtain value being input. Lines 4 and 6 are badly indented; being part of same function, those should be at the same indentation

level as that of lines 2, 3, 5 and 7. And also, variable favourite-game is an invalid identifier as it contains a hyphen, but it should have been an underscore.

Question 42:
Write a function that:

(i) Asks the user to input a diameter of circle in inches.
(ii) Sets a variable called radius to one half of that number.
(iii) Calculate the area of circle.
(iv) Print the area of circle with appropriate unit.
(v) Return this same amount as the output of the function.

Answer:

```
def area_circle ():
import math
diameter=float (raw_input (“Enter the diameter in Inches))
radius = diameter 12
area_circle_inches = (math.pi) * (radius * * 2)
print “The area of circle, area_circle_ inches
return (area_circle_inches)
```

Question 43:
What would be the output of the follow-ing ? Explain. [CBSE Text Book]
def fl(): n = 44 def f2 (): n = 77
print “value of n”, n () print “value of n”, n
Answer:
value of n
Name Error : name ‘n’ is not defined.

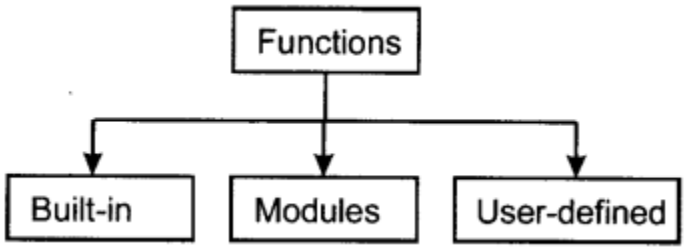
Long Answer Type Questions (4 marks each)

Question 1:
Explain Comments in Python.
Answer:
Comments in Python :
A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the physical line end are part of the comment, and the Python interpreter ignores them.

```
# !/usr/bin/python
# First comment
```

print
“Hello, Python!”
;
second comment
This will produce following result:
Hello, Python!
A comment may be on the same line after a
statement or expression:
name
=
“Madisetti”
This is again comment
You can comment multiple lines as follows :
This is a comment.
This is a comment, too.
This is a comment, too.
I said that already.

Question 2:
What are the various types of functions available in Python ?
Answer:



Built-in functions are those that are built into Python and can be accessed by a programmer, e.g. raw_input, len(s),
A **module** is a file containing Python defintions and statements. We need to import modules to use any of its functions or variables in our code.
e.g. import math
value = math.sqrt (64).
User-defined functions are created by the programmer.
e.g. def area (side):
a = side * side
return a

Question 3:
Explain the term ‘Module’.
Answer:
Module : A module is a file containing Python definitions and statements. Standard library of

Python is extended as modules to a programmer. Definitions from the module can be used within the code of a program. To use these modules in the program, a programmer needs to import the module. There are many ways to import a module in program :
(i) Import
(ii) from
(i) Import: It is simplest and most common way to use modules in our code. Its syntax is :
> > > import module name 1 [, module name 2,…….]
Example,
> > > import math
> > > value = math.sqrt (25)
(ii) From : It is used to get a specific function in the code instead of the complete module file. If we know before hand which functions, we will be needing, then we may use 'from'. For modulus having large number of functions, it is recommended to use 'from' instead of import. Its syntax is
> > > from module name import function name 1 [, function name 2, …….]
Example,
> > > from math import sqrt
> > > value = math . sqrt (36)

Question 4:
Explain ceil(x) in python.
Answer:
ceil(x)
Description : The method ceil( ) returns ceiling value of x-the smallest integer not less than x.
Syntax:
Following is the syntax for ceil()
method import math
math.ceil(x)
Note: This function is not accessible directly so we need to import math module and then we need to call this function using math static object.
Parameters : x—This is a numeric expression.
Return Value : This method returns smallest integer not less than x.
Example:

```
# !/usr/bin/py thon
import math # This will import math module
print "math.ceil(-45.17):",
math.ceil(-45.17)
print "math.ceil(100.12):",
math.ceil(100.12)
print "math.ceil(100.72):",
math.ceil(100.72)
print "math.ceil(119L):",
```

math.ceil(119L)
print “math.ceil(math.pi):”,
math.ceil(math.pi),
This will produce the following result
math.ceil(-45.17) :-45.0
math.ceil(100.12): 101.0
math.ceil(100.72): 101.0
math.ceil(119L): 119.0
math.ceil(math.pi): 4.0

Question 5:
Explain exp(x) in python with example.
Answer:
Description : The method exp( ) returns exponential of x : ex.
Example:
Syntax: Following is the syntax for exp() method
import math
math.exp(x)
Note: This function is not accessible directly so w e need to import math module and then we need to call this function using math static object.
Parameters : x—This is a numeric expression.
Return Value: This method returns exponential of x: ex.
Example: The following example shows the usage of exp() method

# !/usr/bin/py thon
import math # This will import math module
print “math.exp(-45.17): “,
math.exp(-45.17)
print “math.exp(100.12):”,
math.exp(100.12)
print “math.exp(100.72):”,
math.exp(100.72)
print “math.exp(119L):”,
math.exp(119L)
print “math.exp(math.pi):”,
math.exp(math.pi),

This will produce the following result

math.exp(-45.17): 2.41500621326e-20
math.exp(100.12): 3.03084361407e+43
math.exp(100.72): 5.52255713025e+43

math.exp(119L): 4.7978133273e+51
math.exp(math.pi): 23.1406926328

Question 6:
Explain floor(x) in Python with example.
Answer:
The method floor() returns floor of x – the largest integer not greater than x.
Syntax: Following is the syntax for floor() method
import math
math.floor(x)
Note: This function is not accessible directly so we need to import math module and then we need to call this function using math static object.
Parameters : x — This is a numeric expression.
Return Value: This method returns largest integer not greater than x.
Example: The following example shows the usage of floor() method.

```
# !/usr/bin/python
import math # This will import math module
print “math.floor(-45.17): “,
math.floor(-45.17)
print “math.floor(100.12): “,
math.floor(100.12)
print “math.floor(100.72): “,
math.floor(100.72)
print “math.floor(119L):”,
math.floor(119L)
print “math.floor(math.pi):”,
math.floor(math.pi)
```

This will produce the following result:

```
math.floor(-45.17) :-46.0
math.floor(100.12): 100.0
math.floor(100.72): 100.0
math.floor(119L): 119.0
math.floor(math.pi): 3.0
```

Question 7:
Explain log(x) in Python with example.
Answer:
Description : The method log( ) returns natural logarithm of x, for x > 0.
Syntax : Following is the syntax for log() method

import math
math.log(x)
Note: This function is not accessible directly so we need to import math module and then we need to call this function using math static object.
Parameters : x — This is a numeric expression.
Return Value: This method returns natural logarithm of x, for x > 0.
Example: The following example shows the usage of log() method.

```
# !/usr/bin/python
import math # This will import
math module
print “math.log(100.12): ” ,
math.log(100.12)
print “math.log(100.72): “,
math.log(100.72)
print “math.log(119L):”,
math.log(119L)
print “math.log(math.pi):”,
math.log(math.pi)
```

This will produce the following result:

```
math.log(100.12): 4.60636946656
math.log(100.72): 4.61234438974
math.log(119L): 4.77912349311
math.log(math.pi): 1.14472988585
```

Question 8:
Explain loglO(x) in Python with example.
Answer:
Description: The method logl0() returns base -10 logarithm of x for x > 0.
Syntax: Following is the syntax for logl0() method
import math
math.log10(x)
Note: This function is not accessible directly so we need to import math module and then we need to call this function using math static object.
Parameters : x — This is a numeric expression.
Return Value : This method returns base -10 logarithm of x for x > 0.
Example: The following example shows the usage of log10() method.

```
# !/usr/bin/python
import math # This will import math module
```

```
print “math.logl0(100.12): “,
math.logl0(100.12)
print “math.logl0(100.72):”,
math.logl0(100.72)
print “math.loglO(119L):”,
math.log10(119L)
print “math.loglO(math.pi):”,
math.log10(math.pi)
```

This will produce the following result:

```
math.logl0(100.12): 2.00052084094
math.logl0(100.72): 2.0031157171
math.log10(119L): 2.07554696139
math.loglO(math.pi): 0.497149872694
```

Question 9:

Explain pow(x, y) in python with example.

Answer:

Description : The method pow() returns returns the value of x to the power of y.

Syntax: Following is the syntax for pow() method

```
import math
math.pow(x,y)
```

Note : This function is not accessible directly so we need to import math module and then we need to call this function using math static object.

Parameters : x—This is a numeric expression, y—This is also a numeric expression.

Return Value : This method returns value of x to the power of y.

Example: The following example shows the usage of pow() method.

```
# !/usr/bin/python
import math # This will import math module
print “math.pow(100, 2): “,
math.pow(100,2)
print “math.pow(100,-2): “,
math.pow(100,-2)
print “math.pow(2,4): “,
math.pow(2,4)
print “math.pow(3, 0): “,
math.pow(3,0)
```

This will produce the following result:

math.pow(100, 2) :10000.0
math.pow(100,-2): O.OOOl
math.pow(2, 4): 16.0 math.pow(3,0): 1.0

Question 10:
Describe sqrt(x) in Python with example.
Answer:
Description : The method sqrt( ) returns the square root of x for x > 0.
Syntax: Following is the syntax for sqrt() method
import math
math.sqrt(x)
Note: This function is not accessible directly so we need to import math module and then we need to call this function using math static object.
Parameters: x —This is a numeric expression.
Return Value : This method returns square root of x for x > 0.
Example: The following example shows the usage of sqrt() method.

```
# !/usr/bin/python
import math # This will import math module
print "math.sqrt(100): ", math.sqrt(100)
print "math.sqrt(7): ", math.sqrt(7)
print "math.sqrt(math.pi): ", math.sqrt(math.pi)
```

This will produce the following result:

```
math.sqrt(100): 10.0
math.sqrt(7): 2.64575131106
math.sqrt(math.pi): 1.77245385091
```

Question 11:
Describe Random numbers function in Python with example.
Answer:
Description : The method random( ) returns a random float r, such that 0 is less than or equal to r and r is less than 1.
Syntax: Following is the syntax for
random() method
random()
Note : This function is not accessible directly so we need to import random module and then we need to call this function using random static object.
Parameters: NA
Return Value : This method returns a random float r, such that 0 is less than or equal to r and r

is less than 1.
Example: The following example shows the usage of random() method.

```
#!/usr/bin/python
import random # First random number
print “random(): “,
random.random() # Second random number
print “random(): “,
random.random()
```

This will produce the following result:

```
random(): 0.281954791393
random(): 0.309090465205
```

Question 12:
Explain cos() in Python.
Answer:
Description: The method cos() returns the cosine of x radi **Answer:**
Syntax : Following is the syntax for cos() method
cos(x)
Note: This function is not accessible directly so we need to import math module and then we need to call this function using math static object.
Parameters : x—This must be a numeric value.
Return Value : This method returns a numeric value between-1 and 1 which represents the cosine of the angle.
Example: The following example shows the usage of cos() method

```
# !/usr/bin/python
import math
print “cos(3):”,
math.cos(3)
print “cos(-3):”,
math.cos(-3)
print “cos(0):”,
math.cos(0)
print “cos(math.pi):”,
math.cos(math.pi)
print “cos(2*math.pi):”,
math.cos(2*math.pi)
```

This will produce the following result:

cos(3) >0.9899924966
cos(-3) >0.9899924966
cos(0): 1.0
cos(math.pi) >1.0
cos(2*math.pi): 1.0

Question 13:
What are Generator Functions ?
Answer:
A generator function or generator method is one which contains a yield expression. When a generator function is called it returns an iterator. Values are extracted from the iterator one at a time by calling its_next_( ) method. At each call to_next_( ) the generator function's yield expression's value (None if none is specified) is returned. If the generator function finishes or executes return a Stop Iteration exception is raised.
Generators provide an elegant way to write simple and efficient code for functions that return a list of elements. Based on the yield directive, they allow you to pause a function and return an intermediate result. The function saves its execution context and can be resumed later if necessary.

Question 14:
Define functions with example.
Answer:
A function is a block of organized, reusable code that is used to perform a single, related action. Functions provides better modularity for your application and a high degree of code reusing.As you already know, Python gives you many built-in functions like
print( ) etc. but you can also create your own functions. These functions are called user – defined functions.
Syntax:
def functionname(parameters):
"function_docstring"
function_suite return[expression]
By default, parameters have a positional behavior, and you need to inform them in the same order that they were defined.
Example:
Here is the simplest form of a Python function. This function takes a string as input parameter and prints it on standard screen.
def printme(str):
"This prints a passed string into this function"
print str
return

Question 15:
Define functions calling with example.
Answer:
Defining a function only gives it a specific name, specifies the parameters that are to be included in the function, and structures the blocks of code.
Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt. Following is the example to call printme( ) function:

```
# !/usr/bin/py thon
# Function definition
def printme(str):
"This prints a passed string into this function"
print str;
return;
# Now you can call printme function
printme ("I'm first call to user defined function!");
printme ("Again second call to the same function");
```

This will produce the following result:

```
I'm first call to user defined function!
Again second call to the same function
```

Question 16:
How are arguments passed – by reference or by value ?
Answer:
In Python everything is an object and all variables hold references to objects. The values of these references are to the functions. As a result you can not change the value of the reference but you can modify the object if it is mutable. Remember, numbers, strings and tuples are immutable, list and diets are mutable.

Question 17:
How can we pass optional or keyword parameters from one function to another in Python ?
Answer:
Gather the arguments using the * and ** specifiers in the function's parameter list. It gives us positional arguments as a tuple and the keyword arguments as a dictionary. Then we can pass these arguments while calling another function by using

```
# and **
def funl (a, *tup, ** keyword.Arg):
```

keyword Argj'width'j = '23.3c'
fun2 (a, *tup, **keywordArg)

Question 18:
How to generate random numbers in Python ?
Answer:
The standard module random implements a radom number generator. *'
There are also many other in this module, such as :
uniform(a, b) returns a floating point number in the range [a, b].
randint(a, b) returns a random integer number in the range [a, bj.
random() returns a floating point number in the range [0,1].
Following code snippet show usage of all the different functions, evertime it is executed.
import random
i = random.randint (1, 99) # i randomly initialized by integer between range 1 and 99.
j = random randient (1,99) # j randomly initialized by float between range 1 and 99.
k = random.random() # k randomly initialized by float between range 0 and 1

Print ("i.", i)
Print ("j.",j)
Print ("k.", k)

_____
Output:
('i:', 64)
('j:' 701.85008797642115)
('k:': 0.18173593240301023)
Output:
('i:',83)
('j:',56.817584548210945)
('k:'; 0.9946957743038618)

Question 19:
Explain "The returnStatement" with example.
Answer:
The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.
All the above examples are not returning any value, but if you like you can return a value from a function as follows :

#!/usr/bin/py thon
# Function definition is here
def sum(arg1,arg2):
# Add both the parameters and return them.

” total=arg1+arg2
print “Inside the function : “,
total
return total;
Now you can call sum function total=sum(10,20);
print “Outside the function total

This will produce the following result:

Inside the function : 30
Outside the function : 30

Question 20:
Explain Namespaces and Scoping.
Answer:
Variables are names (identifiers) that map to objects. A namespace is a dictionary of variable names (keys) and their corres-ponding objects (values).
A Python statement can access variables in a local namespace and in the global namespace. If a local and a global variable have the same name, the local variable shadows the global variable.
Each function has its own local namespace. Class methods follow the same scoping rule as ordinary functions. Python makes educated guesses on whether variables are local or global. It assumes that any variable assigned a value in a function is local.
Therefore, in order to assign a value to a global variable within a function, you must first use the global statement.The statement global VarName tells Python that VarName is a global variable.
Python stops searching the local namespace for the variable.

Question 21:
How does a user can share global variables across modules ?
Answer:
The canonical way to share information across modules within a single program is to create a special module (called config or cfg). Just import the config module in all modules of application. The module then becomes available as a global name. Because there is only one instance of each module, any change made to the module object get reflected everywhere. For example,

config. py:
x=0
mod.py:
import config
config.x=1
main.py :

```
import config
import mod
print config.x
```

Question 22:
In analog to the example, write a script that asks users for the temperature in F and prints the temperature in C.
Answer:

```
# Program to convert temperature from F into C.
> > > fahrenheit = input (“Entry the temperature
in F : “)
> > > centigrade = (fahrenheit – 32) 8 5/9.0
> > > print “The equivalent temperature in centigrade is”, centigrade
> > > print
```

Question 23:
Define a function ‘Subtract Number (x, y)’ which takes in two numbers and returns the different of the two.
Answer:

```
# Python function which returns the difference of two numbers
> > > a = input (“Enter the first number :”)
> > > b = input (“Enter the second number :”) def Subtract Number (a, b):
> > > print “Subtraction = %d – %d” % (a, b)
> > > return a – b
```

Question 24:
Write a program that takes a number and calculate and display the log, square, sin and cosine of it.
Answer:

```
# Program to display the various values of a number import math
> > > number = input (“Enter the number :”)
> > > print “Number is”, number
> > > print “Log value of number is” math.log (number)
> > > print “Square of number is”, math.pow (number, 2)
> > > print “Sin value of number is”, math.sin (number)
> > > print “Cosine value of number is”, math.cos (number)
```

Question 25:
Write a tic-tac-toe program in python. [CBSE Text Book]
Answer:

```
# tic-tac-toe program
import random def drawboard (board):
# This function is used to print the board
print (‘ | | ‘)
print (‘ ‘ + board [7] + ‘ j ‘ + board [8]
+ ‘ | ‘ + board [9]) print (‘| | ‘)
print (‘ ‘)
print (‘| | ‘)
print (‘ ‘ + board [4] + ‘ | ‘ + board [5]
+ ‘ | ‘ + board [6]) print (‘| | ‘)
print (‘ ‘)
Print (‘ | | ‘)
print (‘ ‘ + board [1] + ‘ | ‘ + board [2]
+ ‘ | ‘ + board [3]) print (‘| | ‘)
def input player letter ():
# This function is used to display the user typed letter
Letter = ”
while not (letter = = ‘X’ or letter = = ‘O’):
print (‘Do you want to be X or O ?’)
letter = input () .upper ()
if letter = ‘X’ return [‘X’, ‘O’]
else:
return [‘O’, ‘X’] def whogoesFirst ():
# This function is used to randomly choose
the player
if random . randint (0,1) = = 0 : return ‘Computer’ else:
return ‘Player’ def Play Again ():
# This function is used when player wants play again print
(‘Do you want to play again ? (Yes or no)’) return input (), lower ().startswith (‘y’) def
makeMove (board, letter, move): board[move] = letter def is Winner (bo, le):
# This function returns true when plays has
won return


(60[7] = = le and 60[8] = = le and 60[9] = = le) or
(60[4] = = le and 60[5] = = le and 60[6] = = le) or
(60[1 ] = = le and 60[2] = = le and 60[3] = = le) or
(60[7] = = le and 60[4] = = le and 60[1] = = le) or
(60[8] = = le and 60[5] = = le and 60[2] = = le) or
(60[9] = = le and 60[6] = = le and 60[3] = = le) or
```

```
(60[7] = = le and 60[5] = = le and 60[3] = = le) or
(60[9] = = le and 60[5] = = le and 60[1] = = le) or


def getBoardcopy(board):
# This function make a duplicate of the board dupeBoard = [ ]
for i in board :
dupeBoard.append(i)
return depeBoard
def isSpaceFree (board, move):
# This function returns Ttue if the passed
move is free return boardfmove] = = ‘ ‘ def getPlayerMove(board):
# This function is used to take player’s move
move = ‘ ‘
while move not in’12345678 9’.split () or
not is spaceFree (board, int(move)):
print (‘What is your next move ? (1-9)’)
move = input () return int(move)
def chooseRandomMoveFromList(board,
movelist):
# This function returns a valid move from the passed list possible Moves = [ ]
for i in moveList:
if is SpaceFree (board, i):
possibleMOves.append(i)
if len(possibleMoves)! = 0 :
return random.choice (possibleMoves)
else:
return – None
def getComputerMove (board, computerletter):
# This function returns the computer’s letter
if computer letter = = ‘X’:
player letter = ‘O’
else:
player letter = ‘X’
for i in range (1,10):
copy = getBoardCopy (board)
if isSpaceFree (copy, i):
makeMove (copy, computerletter, i)
if is Winner (copy, computerletter):
return i
for i in range (1,10):
copy = getBoard Copy (board)
if isSpaceFree (copy, i):
makeMove (copy, player letter, i)
```

```
if is Winner (copy, player letter):
return i
move = chooseRandomMoveFromList (board, [1, 3, 7, 9])
if move ! = None :
return move
if isSpaceFree (board, 5):
return 5
return ChooseRandomMoveFromList (board, [2,4,6,8])
def is BoardFull (board):
# This function returns true if all space on board has been taken
for i in range (1,10):
if isSpaceFree (board, i):
return False
return True
print (“Welcome to Tic-Tac-Toe !”)
while True:
the Board = [” * 10
playerletter, computerletter = input
playerletter()
turn = whoGoesFirst()
print (‘The’ + turn + ‘will go first.’)
gamelsPlaying = True
while gemelsPlaying:
if turn = = ‘player’:
drawBoard (theBoard)
move = getPlayerMove (theBoard)
makeMove (theBoard, playerletter, move)
if is Winner (theBoard, playletter):
drawBoard (theBoard)
print (‘Hooray ! You have won !’)
gamelsPlaying = False
esle:
if is BoardFull (theBoard):
drawBoard (theBoard)
print (‘The game is a tie.”)
break
else:
turn = ‘computer’
else:
move=getCompterMove (theBoard, Computerletter)
makeMove (theBoard, computer letter, move)
if is Winner (theBoard, computerletter): drawBoard (theBoard)
print(‘The computer has beaten you!’) gamelsPlaying = False
else:
```

```
if isBoardFull (theBoard):
drawBoard (theBoard)
print (‘The game is a tie !’)
break
else:
turn = ‘player’ if not play Again ():
break
```