1a)	Write the following in Python(i) a is greater than any one of x,y,z (ii) (logx+sin45)/xy
Ans	(i)a>x or a>y or a>z (ii) (math.log/x,2)+math.sin/math.radious(45)/(x*Y)
b)	Explain how the following are used in python (i) Multiline String,(ii) Quotes, (iii) Escape character sequence.

Ans	(i) Multiline String: If you create a string using single or double quotes, the whole string
	must fit onto a single line. Here's what happens when you try to stretch a string across multiple
	lines:
	>>> 'one
	Traceback (most recent call last):
	File "<string>" , line 1, in <string>
	Could not execute because an error occurred:
	EOL while scanning single-quoted string: <string>, line 1, pos 4:
	'one
	EOL stands for "end of line," so in this error report, Python is saying
	that it reached the end of the line before it found the end of the string.
	To span multiple lines, put three single quotes or three double quotes
	around the string instead of one of each. The string can then span as
	many lines as you want:
	>>> " 'one
	... two
	... three'"
	' one\ntwo\nthree'
	Notice that the string Python creates contains a \n sequence everywhere
	our input started a new line. In reality, each of the three major
	operating systems uses a different set of characters to indicate the end
	of a line. This set of characters is called a *newline*. On Linux, a newline
	is one '\n' character; on Mac OS X, it is one '\r'; and on Windows, the
	ends of lines are marked with both characters as '\r\n'.
	Python always uses a single \n to indicate a newline, even on operating
	systems like Windows that do things other ways. This is called *normalizing*
	the string; Python does this so that you can write exactly the same
	program no matter what kind of machine you're running on.
	(ii) Quotes: In Python, we indicate that a value is a string by putting either single or double quotes around it:
	>>> 'Aristotle'
	'Aristotle'
	>>> "Isaac Newton"
	'Isaac Newton'
	The quotes must match:
	>>> 'Charles Darwin"
	File "<stdin>" , line 1
	'Charles Darwin"
	^
	SyntaxError: EOL while scanning single-quoted string
	We can join two strings together by putting them side by side:
	>>> 'Albert' 'Einstein'
	'AlbertEinstein'
	Notice that the words Albert and Einstein run together. If we want a space
	between the words, then we can add a space either to the end of Albert
	or to the beginning of Einstein:
	>>> 'Albert ' 'Einstein'

	'Albert Einstein'
	>>> 'Albert' ' Einstein'
	'Albert Einstein'
	It's almost always clearer to join strings with +. When + has two string operands, then it is referred to as the *concatenation operator*:
	>>> 'Albert' + ' Einstein'
	'Albert Einstein'
	(iii) Escape Sequence Character:
	Suppose you want to put a single quote inside a string. If you write it directly, Python will complain:
	
	>>> 'that' s not going to work'
	File "<stdin>" , line 1
	'that' s not going to work'
	^
	SyntaxError: invalid syntax
	The problem is that when Python sees the second quote—the one that you think of as being part of the string—it thinks the string is over. It then doesn't know what to do with all the stuff that comes after the second quote.
	One simple way to fix this is to use double quotes around the string:
	
	>>> "that's better"
	"that's better"
	Escape Sequence Description
	\n End of line
	\\ Backslash
	\' Single quote
	\" Double quote
	\t Tab
	Figure 3.1: Escape sequences
	If you need to put a double quote in a string, you can use single quotes around the string. But what if you want to put both kinds of quote in one string? You could do this:
	>>> 'She said, "That' + '"' + 's hard to read."'
	Luckily, there's a better way. If you type the previous expression into Python, the result is as follows:
	'She said, "That\'s hard to read."'
	The combination of the backslash and the single quote is called an *escape sequence*. When Python sees a
	backslash inside a string, it means that the next character represents something special—in this case, a single quote, rather than the end of the string. The backslash is called an *escape character*, since it signals the start of an escape sequence.

2	Discuss the usage of the following with respect to the print() function: i) sep argument ii) end argument iii) .format(arguments) Explain input() Function.
Ans	

```
print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

Print function is used to output a value /expression on the screen. E.g.

```
>>> print(1 + 1)
2
>>> print("The Latin 'Oryctolagus cuniculus' means 'domestic ra
The Latin 'Oryctolagus cuniculus' means 'domestic rabbit'.
```

"sep" is used to specify the separators between different elements when a list is printed. The default value is a space. example

```
>>> print('a', 'b', 'c')  # The separator is a space by defaul
a b c
>>> print('a', 'b', 'c', sep=', ')
a, b, c
```

"end" is used to specify the character to be printed at the end of list. The default value is newline.
Example

```
>>> print('a', 'b', 'c', sep=', ', end='')
a, b, c>>>
```

input ( ) : This function first takes the input from the user and then evaluates the expression, which means Python automatically identifies whether user entered a string or a number or list. If the input provided is not correct then either syntax error or exception is raised by python. For example –

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

```
Enter your value: 123
123
>>>
```

How the input function works in Python :

- When input() function executes program flow will be stopped until the user has given an input.
- The text or message display on the output screen to ask a user to enter input value is optional i.e. the prompt, will be printed on the screen is optional.
- Whatever you enter as input, input function convert it into a string. if you enter an integer value still input() function convert it into a string. You need to explicitly convert it into an integer in your code using typecasting.

3	Explain the two ways to use python interpreter. What are the errors that can be detected by Python? Differentiate between them with one example each. Give output of the following expressions: i) 54/17 ii) -17/10 iii) -16%5 iv) 17%-10

Ans	There are two ways to use the Python interpreter, one is to tell it to execute a python program that is saved in a file with a .py extension, Another is to interact with it in a program called a shell, where you type statements one at a time. The interpreter will execute each statement when you type it, do what the statement says to do, and show any outpu as text, all in one window. There are two kinds of errors in Python: • syntax errors -coding errors which dont follow Python syntax rules. E.g. c=a b- Here there is no operator between identifiers a and b. • semantic errors: When you tell Python to do something that it cannot do, Example 1 - divide a number by 0 Example 2 - try to use a variable that does not exist.. Example 3 - access element outside array boundary 54/17 - 3, -17/10 - -1.7, -16%5 = 4, 17%-10 = -3
4	Write a program to read marks of 3 test M1, M2,M3 and find the average of best two tests rounding to next integer in case of fraction in average without using if statement.
Ans	m1=int(input()) m2=int(input()) m3=int(input()) s=sum([m1,m2,m3]) best=s-min(m1,m2,m3) avg=int(best/2 +0.5) print("Average of two better marks is", avg)
5	Describe briefly the process of designing your own modules with following example: I module USD to INR conversion and II module INR to USD conversion. Import this module and convert the Indian rupees to USD. Assume 1 USD=72INR.
Ans	Describe briefly the process of designing your own modules with clear example. Sol: Module is a collection of related function and variables. E.g. math module Python allows us to create a module by creating a python file and including definitions of all functions and related variables. the module can then be imported by using import <filename> of file containing the module. Example if we create a file Integer.py

```
x=10
y=20
def add(a,b):
    return a+b

def sub(a,b):
    return a-b

def mult(a,b):
    return a-b
```

Importing the module using
>>> import Integer
results in the function add, mult and sub along with the variables x and y to be available.
They can be used in the following manner:
>>> Integer.add(5,7)

	12
	>>>Integer.mult(3,x)
	30
	
	def INR2USD(x):
	y=x/72
	return y
	def USD2INR(x):
	y=x*72
	return y
	/// Save above file with conversion.py
	import conversion
	
	USD=conversion.INR2USD(2000)
	INR=conversion.USD2INR(50)
	Print("Amount in dollor", USD)
	Print("Amount in Rs.", INR)

6 a)	Evaluate the following expression (i) 5//3*2-6/3*5%3 (ii) 5%8 *3+8%3*5
Ans	(i) 5//3*2-6/3*5%3 2-1=1 (ii) 5%8*3+8%3*5 15+10=25
b)	Explain any 4 string functions with examples.
Ans	swapcase(...) | S.swapcase() -> string | | Return a copy of the string S with uppercase characters | converted to lowercase and vice versa strip(...) | S.strip([chars]) -> string or unicode | | Return a copy of the string S with leading and trailing | whitespace removed. | If chars is given and not None, remove characters in chars instead. | startswith(...) | S.startswith(prefix[, start[, end]]) -> bool | | Return True if S starts with the specified prefix, False otherwise. | With optional start, test S beginning at that position. | With optional end, stop comparing S at that position. | prefix can also be a tuple of strings to try. | split(...) | S.split([sep [,maxsplit]]) -> list of strings | | Return a list of the words in the string S, using sep as the | delimiter string. If maxsplit is given, at most maxsplit | splits are done. If sep is not specified or is None, any | whitespace string is a separator and empty strings are removed | from the result.

7	Write a program to input x and y coordinates of a point. Find the lines in any of the following(i) Origin (ii) x- axis, (iii) y-axis (iv) Ist quadrant (v) II quadrant (vi) III quadrant (vii) IV quadrant

Ans	def quadrant(x, y):
	  if (x > 0 and y > 0):
	  print ("lies in First quadrant")
	
	  elif (x < 0 and y > 0):
	  print ("lies in Second quadrant")
	
	  elif (x < 0 and y < 0):
	  print ("lies in Third quadrant")
	
	  elif (x > 0 and y < 0):
	  print ("lies in Fourth quadrant")
	
	  elif (x == 0 and y > 0):
	  print ("lies at positive y axis")
	
	  elif (x == 0 and y < 0):
	  print ("lies at negative y axis")
	
	  elif (y == 0 and x < 0):
	  print ("lies at negative x axis")
	
	  elif (y == 0 and x > 0):
	  print ("lies at positive x axis")
	
	  else:
	  print ("lies at origin")
	
	x = 1
	y = 1
	quadrant(x, y)
	
	Output: lies in First quadrant

8	Explain how code in Python is tested semi-automatically. What are the two ways of importing a module? Which one is more beneficial? Explain.

Ans	Python has a module called doctest that allows to run tests that are included in the docstring all at once. The function that enables us to print such a report is testmod. It reports on whether the function calls return what we expect. It gives messages on how many tests succeeded and how many failed. The test cases can be specified in the docstring for a function as shown in the example below:
	
	def large(a,b,c):
	  '''
	  (number, number, number) --> number
	
	  Finds the largest of the three numbers given as input
	  >>> large(5,1,3)
	  5
	  >>> large(4,10,7)
	  10

```
        '''
        if a>b:
                if a>c:
                        return a
                else:
                        return c
        else:
                if  b > c:
                        return b
                else:
                        return c
```

In the above function to find maximum of 3 numbers the docstring specifies two tests for the numbers 5,1,3 and 4,10,7. The line next to the function call in docstring specifies the output expected. The testmod function parses the docstring runs the test specified and compares the result to the expected result to give the output. For instance importing the module containing the function above and typing the following commands in the python

```
>>>import doctest
>>> doctest.testmod()
```

tests all the functions in the current enviroment. In this case it will give the following output:

```
>>> doctest.testmod()
TestResults(failed=0, attempted=2)
```

This means two tests were run and none of them failed.