TYPE B

SHORT ANSWER QUESTIONS [2, 3 Marks]

Non-Programming Questions

182. *How many ways are there in Python to represent an integer literal ?*

Ans. Python allows *three* types of integer literals :

(*a*) Decimal (base 10) integer literals

(*b*) Octal (base 8) integer literals

(*c*) Hexadecimal (base 16) integer literals

(*a*) **Decimal Integer Literals.** An integer literal consisting of a sequence of digits is taken to be decimal integer literal unless it begins with 0 (digit zero).

For instance, 1234, 41, +97, −17 are decimal integer literals.

(*b*) **Octal Integer Literals.** A sequence of digits starting with 0 (digit zero) is taken to be an octal integer.

For instance, decimal integer 8 will be written as 010 as octal integer. ($8_{10} = 10_8$) and decimal integer 12 will be written as 014 as octal integer ($12_{10} = 14_8$).

(*c*) **Hexadecimal Integer Literals.** A sequence of digits preceded by ***0x*** or ***0X*** is taken to be an hexadecimal integer.

For instance, decimal 12 will be written as 0XC as hexadecimal integer.

Thus number 12 will be written either as 12 (as decimal), or as 014 (as octal) or as 0XC (as hexadecimal).

183. *The following code is not giving desired output. We want to input value as 20 and obtain output as 40. Could you pinpoint the problem ?*

```
Number = input( "Enter Number" )
DoubleTheNumber = Number * 2
Print (DoubleTheNumber)
```

Ans. The problem is that **input( )** returns value as a string, so the input value 20 is returned as string '20' and not as integer 20. So the value of variable ***DoubleTheNumbers*** is 2020 in place of required output 40.

Also **Print** is not legal statement of Python ; it should be **print**.

184. *Why is following code giving errors ?*

```
name = "Rehman"
print("Greetings !!!")
    print("Hello", name)
    print("How do you do ?")
```

Ans. The problem with above code is inconsistent indentation. In Python, we cannot indent a statement unless it is inside a suite and we can indent only as much as required.

Thus, corrected code will be :

```
name = "Rehman"
print("Greetings !!!")
print("Hello", name)
print("How do you do ?")
```

185. *What are data types ? What are Python's built-in core data types ?*

Ans. The real life data is of many types. So to represent various types of real-life data, programming languages provide ways and facilities to handle these, which are known as *data types*.

Python's built-in core data types belong to :

- Numbers (integer, floating-point, complex numbers, Booleans)
- String
- List
- Tuple
- Dictionary

186. *If you are asked to label the Python loops as determinable or non-determinable, which label would you give to which loop ? Justify your answer.*

Ans. The 'for loop' can be labelled as ***determinable loop*** as number of its iterations can be determined beforehand as the size of the sequence, it is operating upon.

The 'while loop' can be labelled as ***non-determinable loop***, as its number of iterations cannot be determined beforehand. Its iterations depend upon the result of a test-condition, which cannot be determined beforehand.

187. *There are two types of else clauses in Python. What are these two types of else clauses ?*

Ans. The two types of Python else clauses are :

(*a*) else in an if statement (*b*) else in a loop statement

The *else* clause of an *if* statement is executed when the condition of the if statement results into *false*.

The *else* clause of a *loop* is executed when the loop is terminating normally *i.e.*, when its test-condition has gone *false* for a *while* loop or when the *for* loop has executed for the last value in sequences.

188. *What are immutable and mutable types ? List immutable and mutable types of Python.*

[Textbook Q. 15, Chapter 1 (Type A)]

Ans. The immutable types are those that can never change their value in place. In Python, the following types are immutable : *integers, floating point numbers, Booleans, strings, tuples.*

The mutable types are those whose values can be changed in place. Only three types are mutable in Python, which are : *lists, dictionaries* and *sets.*

189. *What is the difference between implicit type conversion and explicit type conversion ?*

[Textbook Q. 16, Chapter 1 (Type A)]

Ans. An implicit type conversion is a conversion of data type of a value, performed by the compiler without programmer's intervention. This takes place when an expression contains values of two different data types in it.

The explicit type conversion of an operand to a specific type is called type casting. The explicit type conversion is forced by programmer using the typecast operator ( ).

For example,

```
A = 5
B = 7.5
C = A + B          # implicit conversion will take place here as A and B are
                     of different types
D = A + (int)B     # explicit type conversion of B has taken place here
```

190. *An immutable data type is one that cannot change after being created. Give three reasons to use immutable data.*

[Textbook Q. 17, Chapter 1 (Type A)]

Ans. The immutable types offer following advantages/benefits :

(*i*) Values remain safe as values cannot be changed in place.

(*ii*) Immutable types are more suited for things where values must remain intact, such as keys of a dictionary.

(*iii*) Immutable types internally give a better (optimised) performance as value change overheads are not there.

191. *Explain the use of the pass statement. Illustrate it with an example.*

[Textbook Q. 19, Chapter 1 (Type A)]

Ans. In Python , the **pass** statement is an empty statement as it does nothing. Wherever Python encounters a pass statement, Python does nothing and moves to next statement in the flow of control.

For example, in the following loop, for every value divisible by 2, Python will do nothing because of pass statement.

For other values, it will print the value of variable **val**.

```
for val in range(3, 9) :
    if val %2 == 0 :
        pass
    print(val)
```

192. *In the below given code fragments, indicate the data type of each coloured part by choosing the correct type of data from the following type.*

(*a*) int (*b*) float (*c*) bool (*d*) str (*e*) function (*f*) list of int (*g*) list of str

[Textbook Q. 20, Chapter 1 (Type A)]

(*i*)
```
L = inputline.split( )
while L != ( ) :
    print(L)
    L = L[1 :]
```
it converts the read line into words

(*ii*)
```
L = [ 'Hiya', 'Zoya', 'Preet' ]
print(L[0] + L[1] )
```

Ans. (*i*) List (*ii*) String

193. *Fill in the missing lines of code in the following code. The code reads in a limit amount and a list of prices and prints the largest price that is less than the limit. You can assume that all prices and the limit are positive numbers. When a price 0 is entered the program terminates and prints the largest price that is less than the limit.* **[Textbook Q. 1, Chapter 1 (Type B)]**

```
#Read the limit
limit = float(input("Enter the limit"))
max_price = 0
# Read the next price
next_price = float(input("Enter a price or 0 to stop:"))
while next_price > 0 :
    <write your code here>
    #Read the next price
    <write your code here>
if max_price > 0:
    <write your code here>
else :
    <write your code here>
```

Ans.

```
#Read the limit
limit = float(input("Enter the limit"))
max_price = 0
# Read the next price
next_price = float(input("Enter a price or 0 to stop:"))
while next_price > 0 :
    if next_price < limit and next_price > max_price :
        max_price = next_price
    #Read the next price
    next_price = float(input("Enter a price or 0 to stop:"))
if max_price > 0:
    print("Largest price less than the limit is", max_price)
else :
    print("No price less than the limit was entered")
```

194. *Predict the outputs of the following programs :* **[Textbook Q. 2, Chapter 1 (Type B)]**

(*a*)
```
count = 0
while count < 10:
    print('Hello')
    count += 1
```

(*b*)
```
x = 10
y = 0
while x > y:
    print(x, y)
    x = x - 1
    y = y + 1
```

(*c*)
```
keepgoing = True
x = 100
while keepgoing:
    print(x)
    x = x - 10
    if x < 50:
        keepgoing = False
```

(*d*)
```
x = 45
while x < 50:
    print(x)
```

(*e*)
```
for x in [1,2,3,4,5]:
    print(x)
```

(*f*)
```
for p in range(1,10):
    print(p)
```

(*g*)
```
for z in range(-500,500,100):
    print(z)
```

(*h*)
```
x = 10
y = 5
for i in range(x - y * 2):
    print("%", i)
```

(*i*)
```
c = 0
for x in range(10):
    for y in range(5):
        c += 1
print(c)
```

(*j*)
```
x = [1,2,3]
counter = 0
while counter < len(x):
    print(x[counter] * '%')
    for y in x:
        print(y * '* ')
    counter += 1
```

(*k*)
```
for x in 'lamp':
    print(str.upper(x))
```

(*l*)
```
x = 'one'
y = 'two'
counter = 0
while counter < len(x):
    print(x[counter], y[counter])
    counter += 1
```

(*m*)
```
x = "apple, pear, peach"
y = x.split(", ")
for z in y :
    print(z)
```

(*n*)
```
x = 'apple,pear,peach,grapefruit'
y = x.split(',')
for z in y:
    if z < 'm':
        print(str.lower(z))
    else:
        print(str.upper(z))
```

[**Note** : No space after comma]

Ans.

(*a*) It prints "Hello" 10 times in separate lines

(*b*)
```
10 0
9 1
8 2
7 3
6 4
```

(*c*)
```
100
90
80
70
60
50
```

(*d*) It is an ENDLESS loop that keeps printing 45

(*e*)
```
1
2
3
4
5
```

(*f*)
```
1
2
3
4
5
6
7
8
9
```

(*g*)
```
-500
-400
-300
-200
-100
0
100
200
300
400
```

(*h*) No output (this code prints nothing)

(*i*) 50

(*j*)
```
%
*
* *
* * *
%%
*
* *
* * *
%%%
*
* *
* * *
```

(*k*)
```
L
A
M
P
```

(*l*)
```
o t
n w
e o
```

(*m*)
```
apple
pear
peach
```

(*n*)
```
apple
PEAR
PEACH
grapefruit
```

195. *Rewrite the following code in python after removing all syntax error(s). Colour and Underline each correction done in the code.* **[CBSE Sample Paper 2019-20]**

```
30 = To
for K in range(0,To)
    IF k%4 == 0:
        print (K * 4)
    Else:
        print (K + 3)
```

Ans.

```
To = 30                          # variable name should be on LHS
for K in range(0,To) :           # : was missing
    if k%4 == 0:                 # it should be in lower case
        print (K * 4)
    else:                        # else should be in lower case
        print (K + 3)
```

196. *Find and write the output of the following python code :* **[TB Q. 3, Chapter 1 (Type B)]**

```
for Name in ['Jayes', 'Ramya', 'Taruna', 'Suraj'] :
    print (Name)
    if Name[0] == 'T' :
        break
    else :
        print ('Finished!')
print ('Got it!')
```

Ans.

```
Jayes
Finished!
Ramya
Finished!
Taruna
Got it!
```

197. *Find the errors in following code and write the correct code.*

(*i*) *Underline the corrections*

(*ii*) *Write the reason/error next to it in comment form.*

(*a*)
```
if v < 5:
    for j in range(v):
        print "ABC"
 else:
    print "XYZ"
```

(*b*)
```
Def s(x):
    a = 'k'
    print(a * x)
    print(a * str(x) )
for in [1, 2', 10 :
 s(n)
```

(*c*)
```
FOR x in [2, 5, 8]
    n = n + x
print(n)
```

(*d*)
```
x = True
y = False
z = false
if x , y ,z  :
    print " yes "
else :
    print " no "
```

(e)
```
l = ['a', 'b', 'c', 'd']
v = 6
for i in len(l):
    l[i] += v
    v = v - 1
    print("v = ", v, "l =", l)
```

(f)
```
l= ['a', 'b', 'c', 'd']
v= '6'
for in l:
    l[i] += v
    v = v - 2
    print("v= ", v, "l =", l)
```

(g)
```
l = ['a', 'b', 'c', 'd']
t = (0, 1, 2, 3)
v1 = 6
v2 = '6'
for i in t:
    l[i] += v2
    v1 = v1 - 1
    print(t * t)
    print(t [ v1 ] )
    print(t [ v2 : ]
```

(h)
```
s = [11, 13, 15]
for n in arange(len(s)):
  tot = tot + s(n)
print(tot)
```

(i)
```
s = [11, 13, 15]
for n in len(s) :
  tot = tot + s[n]
print(tot)
```

(j)
```
def extract_lesser (l, v):
    for num in l:
    if:
        less_list.append(num)
 Return less_list
```

(k)
```
for fizzbuzz in range(1, 15, ):
    if fizzbuzz % 3 = 0 and fizzbuzz % 5 == 0:
        print "fizzbuzz"
        Continue
    else if fizzbuzz % 3 == 0:
        continue
    else fizzbuzz %% 5 == 0:
        print("buzz")
    print(fizzbuzz)
```

Ans.

(*a*)

```
v = 3                          # v must be defined before being used
if v < 5:
    for j in range(v):
        print("ABC")           # () missing for print()
    else:                      # wrong indentation; else clause can either be for if
                               # or for for loop
        print ( "XYZ" )        # () missing for print()
```

(*b*)

```
def s(x):                      # def should be in lowercase
    a = 'k'
    print(a*x)
    print(a + str(x) )         # two strings cannot be used with * operator,
                               # but can use + operator
for n in [1, 2, 10 ]:          # (i) loop variable not defined
                               # (ii) closing ] missing
    s(n)                       # incorrect indentation
```

(*c*)

```
n = 0                          # variable n must be defined before use
for x in [2, 5, 8] :           # for should be in lowercase and : missing at the end of for
    n = n + x
print(n)
```

(*d*)

```
x = True
y = False
z = False                      # either false must be a predefined variable or
                               # use Boolean value False
if x or y and z :              # Boolean values / expressions must be combined using
                               # or/and/not operators to form a condition
    print ( " yes " )          # () missing for print()
else :
    print ( " no " )           # () missing for print()
```

(*e*)

```
l = ['a', 'b', 'c', 'd']
v = 6
for i in range(len(l)):        # len(l) will return an integer not an iterator;
                               # range(len(l)) will return an iterator
    l[i] += str(v)             # l[i] contains a string and an int can't be
                               # added to a string
    v = v - 1
    print("v = ", v, "l =", l)
```

(*f*)

```
l = ['a', 'b', 'c', 'd']
v = '6'
for i in range( len (l ) ) : # (i) loop variable missing
                             # (ii) loop variable i is used as index of list l in
                             # the loop so i must get integer values from
                             # iterator, thus range ( len(L) ) given here
    l[i] += v
    v = v * 2          #- cannot be used with string and integer, but * can be used
    print("v = ", v, "l =", l)
```

(*g*)

```
l = ['a', 'b', 'c', 'd']
t = (0, 1, 2, 3)
v1 = 6
v2 = '6'
for i in t :           # : missing
    l[i] += v2
    v1 = v1 - 1
    print( t + t)      # a tuple cannot be multiplied with a tuple,
    print(t [:v1] )    # v1 is 6, index out of range, however with slice it will work
    print(t [int(v2):] )       # a tuple slice can contain only integers
```

(*h*)

```
s = [11, 13, 15]
tot = 0                    # tot must be defined before being used
for n in range(len(s)):    # built-in function is range() not arange()
    tot = tot + s[n]       #incorrect indentation and
                           # s is a list; to access its elements
                           # square brackets are used
print(tot)
```

(*i*)

```
s = [11, 13, 15]
tot = 0                    # tot must be defined before being used
for n in range(len(s)) :   # len(s) returns integer which is not iterable,
                           # to make an iterable from integer range() is used
    tot = tot + s[n]       #incorrect indentation
print(tot)
```

(*j*)

```
def extract_lesser (l, v):
    less_list = []         #a list must be defined before being appended
    for num in l:
        if num < v :       # (i) wrong indentation, (ii) condition missing with if
            less_list.append(num)
    return less_list       # (i)wrong indentation, (ii) return must be in lowercase
```

(*k*)

```
for fizzbuzz in range(1, 15, ) :
    if fizzbuzz % 3 == 0 and fizzbuzz % 5 == 0:     # comparison operator is ==
        print ( "fizzbuzz" )                        # () missing in print()
        continue                                    # Continue is not a legal command
    elif fizzbuzz % 3 == 0:                         # else if is not a valid combination
        continue
    elif fizzbuzz % 5 == 0:                         # (i) you cannot give condition with else
                                                    # (ii) %% is not a valid operator
        print("buzz")
    print(fizzbuzz)
```

198. (*a*) *How many times will the following for loop execute and what's the output ?*

(*i*)
```
for i in range(-1,7, -2):
    for j in range (3):
        print(1, j)
```

(*ii*)
```
for i in range(1,3,1):
    for j in range(i + 1):
        print('*')
```

[Textbook Q. 4, Chapter 1 (Type B)]

(*b*) *Consider the following code and determine how many times statements 1 and 2 (marked) will get executed ?*

```
for a in range (0, 7, 4) :
    print (a)                       # statement 1
    for b in range (0, a+2, 2)
        print (b)                   # statement 2
```

Ans. (*a*)

(*i*) The loop will execute 0 times and nothing will be printed because range (–1, 7, – 2) will yield empty sequence.

(*ii*) The outer loop will execute 2 times. The inner loop will repeat for 2 times in first iteration of outer loop and 3 times in second iteration of outer loop.

The output produced will be :

```
*
*
*
*
*
```

(*b*) ***Statement 1*** will get executed 2 times as the outer loop repeats twice because **range (0, 7, 4)** yields a sequence **[0, 4]**.

Statement 2 will get executed **4 times** :

- once for outerloop's value 0 **(range (0, 2, 2)** will give [0]
- thrice for outerloop's value 4 **(range (0, 6, 2)** will give [0, 2, 4]

199. *Is the loop in the code below infinite ? How do you know (for sure) before you run it ?*

```
m = 3
n = 5
```

```
while n < 10:
    m = n - 1
    n = 2 * n - m
    print(n, m)
```

[Textbook Q. 5, Chapter 1 (Type B)]

Ans. No, the loop in not infinite BECAUSE the loop control variable's value is getting updated inside the loop body with a value that will make it reach the condition sometime.

The output produced will be :

```
6 4
7 5
8 6
9 7
10 8
```

200. *What is indexing in context to Python strings ? Why is it also called two-way indexing ?*

Ans. In Python strings, each individual character is given a location number, called index and this process is called indexing. Python allocates indices in two directions :

- in forward direction, the indexes are numbered as 0, 1, 2,.... length-1.
- in backward direction, the indexes are numbered as –1, –2, –3.... length.

This is known as two-way indexing.

201. *What is a string slice ? How is it useful ?*

Ans. A sub-part or a slice of a string, say **s**, can be obtained using **s [n : m]** where *n* and *m* are integers. Python returns all the characters at indices *n*, *n* + **1**, *n* + **2**...*m* – **1** *e.g.*,

```
'Well done'[1 : 4]    will give    'ell'
```

202. *Write a Python script that traverses through an input string and prints its characters in different lines – two characters per line.*

[Textbook Q. 2, Chapter 2 (Type A)]

Ans.

```
name = "String"
size = len(name)
for i in range(0, size, 2) :
    print(name[i], name[i+1])
```

203. *Following code is not producing any output and seems to have gone in endless loop :*

```
lst = ['ab', 'cd']
for i in lst:
    lst.append(i.upper())
print(lst)
```

(*a*) *What could be the reason ?*

(*b*) *Could you suggest a solution for above code problem ?*

Ans. (*a*) Initially the list x contains 2 elements and with every pass of the loop, an element gets added to ***lst*** and the loop has to repeat till the last element of the list. Since list keeps on increasing in size in each loop iteration, the loop will never end, hence the endless loop.

(*b*) If we can limit the loop iterations by clearly specifying the last element for loop iteration, the loop will end after that, *e.g.*, one possible solution could be :

```
lst = ['ab', 'cd']
for i in lst[ :2] :
    lst.append(i.upper())
print(lst)
```

204. *What is the output of the following ?*

```
L = ['im', 'ur']
for i in L:
    print(i.upper())
print(L)
```

Ans.
```
IM
UR
['im', 'ur']
```

205. *What is the output of the following ?*

```
i = 9
while True:
    if (i+1 )% 4 == 0:
        break
    print(i, end = ' ')
    i += 1
```

Ans. 9 10

206. *What is the output of the following ?*

```
i = 4
while True:
    if i% 0O7 == 0:
        break
    print(i, end = ' ')
    i += 1
```

Ans. 4 5 6 (**Note.** 0o7 is a valid integer, written in octal forming)

207. *What is the output of the following ?*

```
i = 4
while True:
    if i% 0o8 == 0:
        break
    print(i, end = ' ')
    i += 1
```

Ans. The above code will give **Error** because 0o8 is not a valid integer. Any number beginning with 0 is treated as octal number and octal numbers cannot have digits 8 and 9. Hence 0o8 is an invalid integer.

208. *What is the output of the following ?*

```
True = False
while True:
    print(True)
    break
```

Ans. The above code will give **Error** because keyword True has been used as variable (in first line of code **True = False**). We cannot use keywords as variables or any other identifiers.

209. *What is the output of the following ?*

(*a*)
```
x = 'abcd'
for i in x:
    print(i, end = ' ')
    i.upper()
```

(*b*)
```
x = 'abcd'
for i in x:
    print(i.upper(), end = ' ')
```

Ans. (*a*) a b c d (*b*) A B C D

210. *What is the output of the following ?*

(*a*)
```
x = 'abcd'
for i in range(len(x)):
    print(i, end = ' ')
```

(*b*)
```
x = 'abcd'
for i in range(len (x)) ;
print(x[i], end = ' ')
```

Ans. (*a*) 0 1 2 3 (*b*) a b c d

211. *What is the output of the following ?*

```
x = 12
for i in x:
    print(i)
```

Ans. The above code will produce error as x is an integer and is used in place of a sequence or iterable in a for loop; hence the error integer values cannot be used as iterables.

212. *What will be the output produced by following code fragments ?* **[TB Q. 1, Chap. 2 (Type B)]**

(*a*)
```
y = str(123)
x = "hello" * 3
print(x, y)
x = "hello" + "world"
y = len(x)
print(y, x)
```

(*b*)
```
x = "hello" +\
    "to Python" +\
    "world"
for char in x :
    y = char
    print(y, ':', end = " ")
```

(*c*)
```
x = "hello world"
print(x[:2], x[:-2], x[-2:])
print(x[6], x[2:4])
print(x[2:-3], x[-4:-2])
```

Ans.

(*a*)
```
hellohellohello 123
10 helloworld
```

(*b*)
```
h :.e :.l :.l :.o :.t :.o :.    :.P :.y :.t :.h :.o :.n :.w :.o :.r :.l :.d :.
```

(*c*)
```
he hello wor ld
w ll
llo wo or
```

213. *Write a short Python code segment that adds up the lengths of all the words in a list and then prints the average (mean) length.* **[Textbook Q. 2, Chapter 2 (Type B)]**

Ans.
```
Line = "This is going to be fun"
Lst = Line.split()
total = 0
```

```
count = 0
for w in Lst:
    count = count + 1
    lenword = len(w)
    total = total + lenword
avg = total/count
print("Original line :", Line)
print("List of words :", Lst)
print("Total length of all words :", total)
print("Average length of all words :", avg)
```

214. *Predict the output of the following code snippets ?* **[Textbook Q. 3, Chapter 2 (Type B)]**

```
a = [1,2,3,4,5]
print(a[3:0:-1])
```

Ans. `[4, 3, 2]`

215. *Predict the output of the following code snippets ?* **[Textbook Q. 4, Chapter 2 (Type B)]**

(*a*)
```
arr = [1, 2, 3, 4, 5, 6]
for i in range(1, 6):
    arr[i - 1] = arr[i]
for i in range(0, 6):
    print(arr[i], end = " ")
```

(*b*)
```
Numbers = [9, 18, 27, 36]
for Num in Numbers :
    for N in range(1, Num%8) :
        print(N, "#", end = " ")
    print( )
```

Ans.

(*a*) `2 3 4 5 6 6`

(*b*)
```
1 # 1 # 2 # 1 # 2 # 3 #
```

216. *Find the errors. State reasons.* **[Textbook Q. 5(*e*), Chapter 2 (Type B)]**

```
for Name in [Amar, Shveta, Parag]
    IF Name[0] = 'S' :
        print(Name)
```

Ans. *Errors :*

(*i*) The ***for loop*** is not created as suite (: **missing** at the end of for loop)

(*ii*) Values ***Amar, Shveta, Parag*** are not enclosed in quotes; also there are no variables defined with these names ; hence they cause error.

(*iii*) **IF** is not a valid statement in Python. It should be **if**.

217. *Assuming words is a valid list of words, the program below tries to print the list in reverse. Does it have an error ? If so, why ? (Hint. There are two problems with the code.)*

```
for i in range(len(words), 0, -1):
    print(words[i], end = ' ')
```

[Textbook Q. 6, Chapter 2 (Type B)]

Ans.

(*i*) The statement **for i in range(len(words), 0, –1)** will give error because, **words[i]** will access the first value as **words[len(words)]** and it is not a valid syntax for the list words.

Valid indexes are from **0 to len(words) –1**

(*ii*) If we correct the above line of code as **for i in range(len(words) –1 , 0, –1)**, still it will not print the first word as ending index given in expression **range(len(words) – 1 , 0, – 1)** and ending index is not included in the range( ) result. The corrected statement should be **for i in range(len(words) –1, –1, –1)** :

218. *What would be the output of following code if ntpl = ("Hello", "Nita", "How's", "life ?") ?*

```
(a, b, c, d) = ntpl
print("a is:", a)
print("b is:", b)
print("c is:", c)
print("d is:", d)
ntpl= (a, b, c, d)
print(ntpl[0][0]+ntpl[1][1], ntpl[1])
```

[Textbook Q. 7, Chapter 2 (Type B)]

Ans.
```
a is: Hello
b is: Nita
c is: How's
d is: life ?
Hi Nita
```

219. *What will be the output of the following code snippet ?* **[Textbook Q. 10, Chapter 2 (Type B)]**

```
my_dict = {}
my_dict[(1,2,4)] = 8
my_dict[(4,2,1)] = 10
my_dict[(1,2)] = 12
sum = 0
for k in my_dict:
    sum += my_dict[k]
print (sum)
print(my_dict)
```

Ans.
```
30
{(1, 2, 4): 8, (4, 2, 1): 10, (1, 2): 12}
```

220. *Write a method in python to display the elements of list thrice if it is a number and display the element terminated with '#' if it is not a number.* **[Textbook Q. 11, Chapter 2 (Type B)]**

For example, if the content of list is as follows :

```
List = ['41', 'DROND', 'GIRIRAJ', '13', 'ZARA']
```

The output should be
```
414141
DROND#
GIRIRAJ#
131313
ZARA#
```

Ans.
```
List = ['41', 'DROND', 'GIRIRAJ', '13', 'ZARA']
for w in List:
    if w.isdigit():
        print(w*3)
    else:
        print(w+"#")
```

221. *What is the significance of having functions in a program ?*

Ans. Creating functions in programs is very useful. It offers following advantages :

(*i*) *The program is easier to understand.*

Main block of program becomes compact as the code of functions is not part of it, thus is easier to read and understand.

(*ii*) *Redundant code is at one place, so making changes is easier.*

Instead of writing code again when we need to use it more than once, we can write the code in the form of a function and call it more than once. If we later need to change the code, we change it in one place only. Thus it saves our time also.

(*iii*) *Reusable functions can be put in a library in modules.*

We can store the reusable functions in the form of modules. These modules can be imported and used when needed in other programs.

222. *From the program code given below, identify the parts mentioned below :*

```
1.      def processNumber(x):
2.          x = 72
3.          return x + 3
4.
5.      y = 54
6.      res = processNumber(y)
```

Identify these parts : *function header, function call, arguments, parameters, function body, main program*

Ans.

Function header	:	**def processNumber(x) :**	*in line 1*
Function call	:	**processNumber(y)**	*in line 6*
Arguments	:	**y**	*in line 6*
Parameters	:	**x**	*in line 1*
Function body	:	x = 72 return x + 3	*in lines 2 and 3*
Main program	:	y = 54 res = processNumber(y)	*in lines 5 and 6*

223. *Trace the following code and predict output produced by it.*

```
1.  def power(b, p) :
2.      y = b ** p
3.      return y
4.
5.  def calcSquare(x) :
6.      a = power(x, 2)
7.      return a
8.
9.  n = 5
10. result = calcSquare(n) + power(3, 3)
11. print(result)
```

Ans. Flow of execution for above code will be :

$1 \rightarrow 5 \rightarrow 9 \rightarrow 10 \rightarrow 5 \rightarrow 6 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 7 \rightarrow 10 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 10 \rightarrow 11$

The output produced by above code will be :

```
52
```

224. *Trace the flow of execution for following programs :*

(*a*)

```
1   def power(b, p):
2       r = b ** p
3       return r
4
5   def cpower(a):
6       a = a + 2
7       a = power(a, 0.5)
8       return a
9
10  n = 5
11  result = cpower(n)
12  print (result)
```

(*b*)

```
1. def increment(x) :
2.     x = x + 1
3.
4. # main program
5. x = 3
6. print(x)
7. increment(x)
8. print(x)
```

(*c*)

```
1.  def increment(x):
2.      z = 45
3.      x = x + 1
4.      return x
5.
6.  # main
7.  y = 3
8.  print(y)
9.  y = increment(y)
10. print(y)
11. q = 77
12. print(q)
13. increment(q)
14. print(q)
15. print(x)
16. print(z)
```

Ans.

(*a*) $1 \rightarrow 5 \rightarrow 10 \rightarrow 11 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 7 \rightarrow 8 \rightarrow 11 \rightarrow 12$

(*b*) $1 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 1 \rightarrow 2 \rightarrow 8$

Control did not return to function call statement (7) as no value is being returned by increment( )

(*c*) $1 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 13 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 14 \rightarrow 15 \rightarrow 16$

Control did not return to function call statement (13) as its result is not being stored anywhere.

225. *Find and write the output of the following python code :*

[CBSE Sample Paper 2019-20]

```
def Change(P, Q = 30):
    P = P+Q
    Q = P-Q
```

```
    print(P,"#",Q)
    return (P)
R = 150
S = 100
R = Change(R,S)
print(R,"#",S)
S = Change(S)
```

Ans.

```
150 # 50
150 # 100
100 # 70
```

226. *A program having multiple functions is considered better designed than a program without any functions. Why ?* **[Textbook Q. 1, Chapter 3 (Type A)]**

Ans. Program having functions is considered better, because :

(*i*) It makes program handling easier as only a small part of the program is dealt with at a time, thereby avoiding ambiguity.

(*ii*) It reduces program size.

(*iii*) It makes a program more readable and understandable to a programmer thereby making program management much easier.

227. *What all information does a function header give you about the function ?* **[Textbook Q. 2, Chapter 3 (Type A)]**

Ans. A function header tells about :

(*i*) the name of the function, (*ii*) the number of arguments or parameters.

228. *What is the difference between the formal parameters and actual parameters? What are their alternative names? Also, give a suitable Python code to illustrate both.*

Ans. ***Actual Parameter*** is a parameter, which is used in a *function call* statement to send the value from *calling function* to the *called function*. It is also known as ***Argument***.

Formal Parameter is a parameter, which is used in a *function header* of the *called function* to receive the value from *actual parameter*. It is also known as ***Parameter***.

For example,

```
def addEm(x, y, z):
    print(x + y + z)

addEm(6, 16, 26)
```

In the above code, *actual parameters* are **6**, **16** and **26** ; and *formal parameters* are *x*, *y* and *z*.

229. *What is the utility of : (i) default arguments, (ii) keyword arguments ?* **[Textbook Q. 5, Chapter 3 (Type A)]**

Ans. (*i*) The ***default parameters*** are parameters with a default value set to them. This default value is automatically considered as the passed value WHEN no value is provided for that parameter in the function call statement.

Thus default arguments are useful when we want to skip an argument in a function call statement and use the default value for it instead.

(*ii*) The ***keyword arguments*** give complete control and flexibility over the values sent as arguments for the corresponding parameters.

Irrespective of the placement and order of arguments, keyword arguments are correctly matched.

230. *Differentiate between fruitful functions and non-fruitful functions.*
[Textbook Q. 8, Chapter 3 (Type A)]

Ans. The functions that return a value *i.e.*, non-void functions are also known as ***fruitful functions***.

The functions that do not return a value, *i.e.*, void functions are also known as ***non-fruitful functions***.

231. *Can a function return multiple values ? How ?* **[Textbook Q. 9, Chapter 3 (Type A)]**

Ans. Yes, a Python function can return more than one value.

To return multiple values from a function, the return statement should have a comma separated multiple values, *e.g.*, following return statement is returning three values :

```
return 23, (a + b), 3**5
```

232. *What do you understand by local and global scope of variables ? How can you access a global variable inside the function, if function has a variable with same name.* **[CBSE SP 19-20]**

Ans. A global variable is a variable that is accessible globally. A local variable is one that is only accessible to the current scope, such as temporary variables used in a single function definition.

A variable declared outside of all the functions or in global scope is known as global variable. A global variable can be accessed inside or outside of the function where as local variable can be used only inside of the function. If a function has a local variable name as a global variable, then in that function scope, the local variable will hide the global variable with the same name. We can access a global variable having the same name as a local variable by declaring its name with keyword **global**, *e.g.*, as **global A**.

233. *What is the difference between a local variable and a global variable ? Also, give a suitable Python code to illustrate both.* **[Textbook Q. 11, Chapter 3 (Type A)]**

Ans. The differences between a local variable and global variable are as given below :

	Local Variable	*Global Variable*
1.	It is a variable which is declared within a function or within a block	It is a variable which is declared outside all the functions
2.	It is accessible only within a function/block in which it is declared	It is accessible throughout the program

For example, in the following code, ***x***, ***xCubed*** are global variables and ***n*** and ***cn*** are local variables.

```
def cube(n):
    cn = n * n * n
    return cn

x = 10
xCubed = cube(x)
print(x,"cubed is", xCubed)
```

234. (*a*) *What will the following codes print ?*

(*i*)
```
def addEm(x, y, z):
    print(x + y + z)

def prod (x, y, z) :
    return x * y * z

a = addEm(6, 16, 26)
b = prod ( 2, 3, 6)
print(a, b)
```

(*ii*)
```
def addEm(x, y, z):
    K = x + y + z

def prod (x, y, z)
    return x * y * z

a = addEm(6, 16, 26)
b = prod ( 2, 3, 6)
print(a, b)
```

(*b*) *In the part (a)'s code, identify the void and non-void functions. The previous code stores the return values of both void and non-void functions in variables. Why did Python not report an error when void functions do not return a value ?*

Ans. (*a*) (*i*)
```
48
None 36
```
(*ii*)
```
None 36
```

(*b*)
```
Void function        :   addEm( )
Non-void function    :   prod( )
```

In Python, void functions do not return a value; rather they report the absence of returning value by returning *None*, which is legal empty value in Python. Thus, variable ***a*** stores **None** and it is not any error.

235. *Consider below given function headers. Identify which of these will cause error and why ?*

(*i*)
```
def func(a = 1, b):
```
(*ii*)
```
def func(a = 1, b, c = 2):
```
(*iii*)
```
def func(a = 1, b = 1, c = 2):
```
(*iv*)
```
def func(a = 1, b = 1, c = 2, d):
```

Ans. Function headers (*i*), (*ii*) and (*iv*) will cause error because **non-default arguments cannot follow default arguments**.

Only function header (*iii*) will not cause any error.

236. *What are the errors in following codes ? Correct the code and predict output :*

(*a*)
```
total = 0;
def sum( arg1, arg2 ):
    total = arg1 + arg2;
    print("Total :", total)
return total;
sum( 10, 20 );
print("Total :", total)
```

(*b*)
```
def Tot(Number)  #Method to find Total
    Sum = 0
    for C in Range (1, Number + 1) :
        Sum += C
    RETURN Sum
print (Tot[3])   #Function Calls
print (Tot[6])
```

[Textbook Q. 1, Chapter 3 (Type B)]

Ans. (*a*)
- Wrong indentation for **return** statement
- Semicolons should be avoided (3 statements have it)

(*b*)
- Range( ) is not a valid function; it should be **range( )**
- RETURN is not a valid statement. It should be **return**.
- Function calls use parenthesis, *i.e.*, ( ) (not square brackets[ ]) to pass values. Thus Function calls for **Tot( )** would be as :
  Tot(3) and **Tot(6)**

237. *Consider the following code and write the flow of execution for this. Line numbers have been given for your reference.* **[Textbook Q. 2, Chapter 3 (Type B)]**

```
1   def power(b, p):
2       y = b ** p
3       return y
4
5   def calcSquare(x):
6       a = power(x, 2)
7       return a
8
9   n = 5
10  result = calcSquare(n)
11  print(result)
```

Ans.

$1 \rightarrow 5 \rightarrow 9 \rightarrow 10 \rightarrow 5 \rightarrow 6 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 7 \rightarrow 10 \rightarrow 11$

238. *Determine the output of the following :*

(*a*)
```
def power(a, b = 2):
    r = 1
    for i in range(b):
        r = r * a
    return r
print ( power(4) )
print ( power(4, 3) )
```

(*b*)
```
def func( alist ):
    alist = [1,2,3,4]
    print (alist)
    return
mylist = [10,20,30]
func( mylist )
print (mylist)
```

(*c*)
```
def func(text, num):
    while num > 0 :
        num = num - 1
        print(num)
num = 4
func('Hello', num)
print(num)
```

(*d*)
```
def printit(list1):
    m = n = list1[0]
    for a in list1:
        if a < n :
            n = a
        if a > m:
            m = a
    print("Biggest", m, "Smallest", n)
list1 = [2, 13, 11, 15, 6]
printit(list1)
```

(*e*)
```
def calc(numbers):
    res = 0
    for x in numbers:
        res += x
    return res
print(sum((8, 2, 3, 0, 7)))
```

(*f*)
```
def myfunction(str1):
    rstr1 = ' '
    index = len(str1)
    while index > 0:
        if str1[ index - 1 ].isalpha():
            rstr1 += str1[ index - 1]
        index = index - 1
    return rstr1
print(myfunction('1234abcd'))
```

(*g*)
```
def determine(s):
    d = {"UPPER" : 0, "LOWER" : 0}
    for c in s:
        if c.isupper():
            d["UPPER"] += 1
        elif c.islower():
            d["LOWER"] += 1
        else:
            pass
    print ("Original String : ", s)
    print ("Upper case count : ", d["UPPER"])
    print ("Lower case count: ", d["LOWER"])
determine('These are HAPPY Times')
```

(*h*)
```
def unique(l):
    x = []
    for a in l:
        if a not in x:
            x.append(a)
    return x
print(unique(['a','b','r','a','c','a','d','a','b','r','a']))
```

(*i*)
```
def compute(s):
    x = []
    for i in range( len(s)):
        a = s[i]
        b = a.upper()
        if a not in x and b not in x :
            if a > 'p' :
                x.append(a.upper())
            else:
                x.append(a)
    return x
print(compute('abracadabra'))
```

(*j*)
```
def create(s):
    enum = ""
    l = range ( len(s) - 1 )
    for n in l :
        if s[n] == s[n + 1] :
            enum += s[n].upper()
        else :
            enum += s[n]
    return enum
print(create("Excellencee") )
```

(*k*)
```
def evn(l):
    enum = " "
    for n in range( len(l) ):
        if n % 2 == 0:
            enum += l[n]
        elif n % 3 == 0:
            enum += l[n].upper()
    return enum
print(evn([ 'a','b', 'r' , 'a', 'c', 'a', 'd', 'a', 'b', 'r', 'a']))
```

Ans.

(*a*)
```
16
64
```

(*b*)
```
[1, 2, 3, 4]
[10, 20, 30]
```

(*c*)
```
3
2
1
0
4
```

(*d*)
```
Biggest 15 Smallest 2
```

(*e*)
```
20
```

(*f*)
```
dcba
```

(*g*)
```
Original String :  These are HAPPY Times
Upper case count :  7
Lower case count:  11
```

(*h*)
```
['a', 'b', 'r', 'c', 'd']
```

(*i*)
```
['a', 'b', 'R', 'c', 'd']
```

(*j*)
```
ExceLlencE
```

(*k*)
```
arAcdbRa
```

239. *What will be the output of following programs ?*

(*i*)
```
num = 1
def myfunc():
    return num
print(num)
print(myfunc())
print(num)
```

(*ii*)
```
num = 1
def myfunc():
    num = 10
    return num
print(num)
print(myfunc())
print(num)
```

(*iii*)
```
num = 1
def myfunc():
    global num
    num = 10
    return num
print(num)
print(myfunc())
print(num)
```

(*iv*)
```
def display():
    print("Hello", end = ' ')
display()
print("there!")
```

[Textbook Q. 5, Chapter 3 (Type B)]

Ans.

(*i*)
```
1
1
1
```

(*ii*) 1
10
1

(*iii*) 1
10
10

(*iv*) Hello there!

240. *Find and write the output of the following python code :* **[CBSE Sample Paper 19-20]**

```
def fun(s):
    k = len(s)
    m = " "
    for i in range(0,k):
        if(s[i].isupper( )):
            m = m+s[i].lower( )
        elif s[i].isalpha( ):
            m = m+s[i].upper( )
        else:
            m = m+'bb'
    print(m)
fun('school2@com')
```

Ans. SCHOOLbbbbCOM

241. *Which names are local and which are global in the following code fragment?*

```
invaders = "Big names"
pos = 200
level = 1

def play( ) :
    max_level = level + 10
    print(len (invaders ) == 0)
    return max_level
res = play()
print(res)
```

Ans.

Global names : invaders, pos, level, res
Local names : max_level

242. *Write a function that takes a positive integer and returns the one's position digit of the integer.*

Ans.

```
def getOnes(num):
    # return the ones digit of the integer num
    onesDigit = num % 10
    return onesDigit
```

243. *What is wrong with the following function definition ?* **[Textbook Q. 7, Chapter 3 (Type B)]**

```
def addEm(x, y, z):
    return x + y + z
    print("the answer is", x + y + z)
```

Ans. The statement with print( ) is unreachable in above code as it follows the *return* statement. As soon as the return statement gets executed, the control returns from the function and no other statement gets executed.

244. *Consider the code below and answer the questions that follow :*

```
def multiply(number1, number2):
    answer = number1 * number2
    return(answer)
    print(number1,'times', number2, '=',answer )
output = multiply(5,5)
```

(*i*) *When the code above is executed, what gets printed ?*

(*ii*) *What is variable output equal to after the code is executed ?*

[Textbook Q. 10, Chapter 3 (Type B)]

Ans.

(*i*) Nothing gets printed (as print() is after the return statement)

(*ii*) 25

245. *Find the errors in code given below :* **[Textbook Q. 11, Chapter 3 (Type B)]**

(*a*)
```
def minus(total, decrement)
    output = total - decrement
    print(output)
    return (output)
```

(*b*)
```
define check()
    N = input ('Enter N: ')
    i = 3
    answer = 1 + i ** 4 / N
    Return answer
```

(*c*)
```
def alpha (n, string = 'xyz', k = 10) :
    return beta(string)
    return n

def beta (string)
    return string == str(n)

print(alpha("Valentine's Day"):)
print(beta (string = 'true'))
print(alpha(n = 5, "Good-bye") :)
```

Ans.

(*a*) **Syntax error.** Colon (:) missing in the end of function header.

(*b*) **Syntax error.** Keyword to define a function is **def** (not define).

Also, colon (:) missing in the end of function header.

Return is not a valid statement. It should be **return**.

(*c*) No error in function alpha's definition. (Multiple return statements are syntactically legal. But in above code, the second return statement is unreachable).

In function beta( )'s definition, Colon (:) missing in the end of function header.

In __main__ part, the colons at the end of first and third print( ) statements is invalid (not enclosed in quotes)

In third print( ) statement, in the function call of alpha( ), positional argument follows keyword argument, which is a syntax error.

246. *In the following code, which variables are in the same scope ?* **[TB Q. 14, Chapter 3 (Type B)]**

```
def func1( ):
    a = 1
    b = 2
def func2( ):
    c = 3
    d = 4
e = 5
```

Ans.

Variables ***a*** and ***b***, have same scope – function scope of func1( )

Variables ***c*** and ***d***, have same scope – function scope of func2( )

247. *What is the output of following code fragments ?* **[Textbook Q. 17, Chapter 3 (Type B)]**

(*i*)
```
def increment(n):
    n.append([4])
    return n
L = [1, 2, 3]
M = increment(L)
print(L, M)
```

(*ii*)
```
def increment(n):
    n.append([49])
    return n[0], n[1], n[2], n[3]
L = [23, 35, 47]
m1, m2, m3, m4 = increment(L)
print(L)
print(m1, m2, m3, m4)
print(L[3] == m4)
```

Ans.

(*i*)
```
[1, 2, 3, [4]] [1, 2, 3, [4]]
```

(*ii*)
```
[23, 35, 47, [49]]
23 35 47 [49]
True
```

248. *What is a module, package and a library ?*

Ans.

Module. A module is a file with some Python code and is saved with a .py extension.

Package. A package is a directory that contains subpackages and modules in it along with some special files such as __init__.py.

Library. A Python library is a reusable chunk of code that is used in program/ script using import command. A package is a library if it is installable or gets attached to ***site-packages*** folder of Python installation.

The line between a package and a Python library is quite blurred and both these terms are often used interchangeably.

249. *What is a Python module ? What is its significance ?*

Ans. A "module" is a chunk of Python code that exists in its own (.py) file and is intended to be used by Python code outside itself.

Modules allow one to bundle together code in a form in which it can easily be used later.

The Modules can be "imported" in other programs so the functions and other definitions in imported modules become available to code that imports them.

250. *What are docstrings ? How are they useful ?*

Ans. A *docstring* is just a regular Python triple-quoted string that is the first thing in a function body/a module/a class.

When executing a function body (or a module / class), the *docstring* doesn't do anything like comments, but Python stores it as part of the function documentation. This documentation can later be displayed using *help( )* function.

So, even though *docstrings* appear like comments (no execution) but these are different from comments.

251. *What happens when Python encounters an* ***import*** *statement in a program ? What would happen, if there is one more import statement for the same module, already imported in the same program ?*

Ans. When Python encounters an **import** statement, it does the following :

- the code of imported module is interpreted and executed.
- defined functions and variables created in the module are now available to the program that imported module.
- For imported module, a new **namespace** is setup with the same name as that of the module.

Any duplicate import statement for the same module in the same program is ignored by Python.

252. *What would be the output produced by the following code :*

```
import math
import random
print(math.ceil(random.random()))
```

Justify your answer.

Ans. The output produced would be :

```
1.0
```

Reason being that ***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**.

253. *Consider the following code :*

```
import math
import random
print(str( int( math.pow( random.randint(2, 4), 2))), end = ' ')
print(str( int( math.pow( random.randint(2, 4), 2))), end = ' ')
print(str( int( math.pow( random.randint(2, 4), 2))))
```

What could be the possible outputs out of the given four choices ?

(*i*) 2 3 4 (*ii*) 9 4 4 (*iii*) 16 16 16

(*iv*) 2 4 9 (*v*) 4 9 4 (*vi*) 4 4 4

Ans. The possible outputs could be (*ii*), (*iii*), (*v*) and (*vi*).

The reason being that ***randint( )*** would generate an integer between range 2...4 , which is then raised to power 2, so possible outcomes can be any one of these *three* : 4, 9 or 16.

254. *What is the problem in the following piece of code ?*

```
from math import factorial
print(math.factorial(5))
```

Ans. In the "from-import" form of import, the imported identifiers (in this case factorial()) become part of the current local namespace and hence their module's names aren't specified along with the module name.

Thus, the statement should be :

```
print(factorial(5))
```

255. *What is a package ? How is a package different from module ?*

[Textbook Q. 3, Chapter 4 (Type A)]

Ans. A module in python is a **.py** file that defines one or more function/classes which you intend to reuse in different codes of your program. To reuse the functions of a given module, we simply need to import the module using **import** command.

A Python package refers to a directory of Python module(s). This feature comes in handy for organizing modules of one type at one place.

256. *What do you understand by standard library of Python ?*

[Textbook Q. 14, Chapter 4 (Type A)]

Ans. A library is a collection of modules (and packages) that together cater to a specific type of applications or requirements.

Python standard library is the library, which is distributed with Python that contains modules for various types of functionalities. Some commonly used modules of Python standard library are :

math module, ***cmath*** module, ***random*** module, ***statistics*** module, ***Urllib*** module, etc..

257. *What is the procedure to create own library/package in Python ?*

Ans. To create own library or package in Python, we should do the following :

(*i*) Create a package folder having the name of the package/library

(*ii*) Add module files (.py files containing actual code functions) to this package folder

(*iii*) Add a special file namely __init__.py to it (even if the file is empty)

(*iv*) Attach this package folder to site-packages folder of Python installation.

258. *Suppose that after we import the random module, we define the following function called diff in a Python session :* **[Textbook Q. 6, Chapter 4 (Type B)]**

```
def diff():
    x = random.random() - random.random()
    return(x)
```

What would be the result if you now evaluate

```
y = diff()
print(y)
```

at the Python prompt ? Give reasons for your answer.

Ans. Variable y would store a non-zero value.

This is because, during both executions of **random.random( )**, a random number would be generated between 0-1 for first execution and successive execution would result in a different random number. Hence a non-zero result .

259. *What are the possible outcome(s) executed from the following code ? Also specify the maximum and minimum values that can be assigned to variable NUMBER.*

```
STRING = "CBSEONLINE"
NUMBER = random.randint(0, 3)
N = 9
while STRING[N] != 'L' :
    print (STRING[N] + STRING[NUMBER] + '#', end = ' ')
    NUMBER = NUMBER + 1
    N = N - 1
```

(i) ES#NE#IO# (ii) LE#NO#ON# (iii) NS#IE#LO# (iv) EC#NB#IS#

[Textbook Q. 7, Chapter 4 (Type B)]

Ans. Part (*iv*) can be one of the possible outcomes.

Maximum value 3, minimum value 0.

260. *Consider the following code :* **[Textbook Q. 8, Chapter 4 (Type B)]**

```
import random
print(int( 20 + random.random() * 5), end = ' ' )
print(int( 20 + random.random() * 5), end = ' ' )
print(int( 20 + random.random() * 5), end = ' ' )
print(int( 20 + random.random() * 5))
```

Find the suggested output options (i) to (iv). Also, write the least value and highest value that can be generated.

(*i*) 20 22 24 25

(*ii*) 22 23 24 25

(*iii*) 23 24 23 24

(*iv*) 21 21 21 21

Ans. Parts (*iii*), (*iv*) can be one of the possible outcomes.

For expression **int(20 + random.random( ) * 5)** :

Maximum value 24, minimum value 20.

For expression **random.random() * 5** :

Maximum value 4.xxxxxx (< 5.0), minimum value 0.0.

261. *Consider the following code :* **[Textbook Q. 9, Chapter 4 (Type B)]**

```
import random
print(100 + random.randint(5, 10), end = ' ' )
print(100 + random.randint(5, 10), end = ' ' )
print(100 + random.randint(5, 10), end = ' ' )
print(100 + random.randint(5, 10))
```

Find the suggested output options (i) to (iv). Also, write the least value and highest value that can be generated.

(*i*) 102 105 104 105 (*ii*) 110 103 104 105

(*iii*) 105 107 105 110 (*iv*) 110 105 105 110

Ans. Parts (*ii*), (*iii*), (*iv*) can be one of the possible outcomes.

For expression **100 + random.randint(5, 10)** :

Maximum value 110, minimum value 105

For expression **random.randint(5, 10)** :

Maximum value 10, minimum value 5

262. *What are the possible outcome(s) executed from the following code ? Also specify the maximum and minimum values that can be assigned to variable PICKER.*

[Textbook Q. 11, Chapter 4 (Type B)]

```
import random
PICK = random.randint(0, 3)
CITY = ["DELHI", "MUMBAI", "CHENNAI", "KOLKATA"] ;
for I in CITY :
    for J in range(1, PICK) :
        print(I, end = " ")
    print( )
```

(*i*)
```
DELHIDELHI
MUMBAIMUMBAI
CHENNAICHENNAI
KOLKATAKOLKATA
```

(*ii*)
```
DELHI
DELHIMUMBAI
DELHIMUMBAICHENNAI
```

(*iii*)
```
DELHI
MUMBAI
CHENNAI
KOLKATA
```

(*iv*)
```
DELHI
MUMBAIMUMBAI
KOLKATAKOLKATAKOLKATA
```

Ans. Part (*iii*) can be one of the possible outcomes.

Maximum value 3, Minimum value 0

263. *What possible output(s) are expected to be displayed on screen at the time of execution of the program from the following code ? Also specify the maximum values that can be assigned to each of the variables FROM and TO.* **[CBSE Sample Paper 19-20]**

```
import random
AR = [20,30,40,50,60,70];
FROM = random.randint(1,3)
TO = random.randint(2,4)
for K in range(FROM,TO+1):
    print (AR[K],end = "#")
```

(*i*) 10#40#70# (*ii*) 30#40#50#

(*iii*) 50#60#70# (*iv*) 40#50#70#

Ans. (*ii*) 30#40#50# Maximum value
for FROM is 2.x but < 3 and
for TO is 4

264. *What is the output of the following code ?*

```
dnry = {0: 'a', 1: 'b', 2: 'c'}
for i in dnry:
    print(i)
```

Ans.

```
0   1   2
```

265. *(a) What is the output of the following code ?*

```
dnry= {0: 'a', 1: 'b', 2: 'c'}
for x, y in dnry:
    print(x, y)
```

(b) Write corrected code for part (a) so that it prints keys and values both.

Ans.

(*a*) The above code will produce **Error** as we can iterate only on keys in a dictionary. Thus the line :

```
for x, y in dnry:
```

is wrong and it will give error.

(*b*)

```
dnry= {0: 'a', 1: 'b', 2: 'c'}
for x in dnry:
    print(x, dnry[x])
```

266. *What is the output of the following ?*

```
dry = {0: 'a', 1: 'b', 2: 'c'}
for x, y in dry.items():
    print(x, y, end = ' ')
```

Ans.

```
0 a 1 b 2 c
```

267. *What is the output of the following ?*

```
d = {0: 'a', 1: 'b', 2: 'c'}
for x in d.keys():
    print(d[x] , end = ' ')
```

Ans.

```
a b c
```

268. *What is the output of the following?*

```
d = {0: 'a', 1: 'b', 2: 'c'}
for x in d.keys():
    print(x, ":", d[x] , end = '...' )
```

Ans.

```
0: a... 1: b... 2: c
```

269. *What is the output of the following ?*

```
d = {0: 'a', 1: 'b', 2: 'c'}
for x in d.values():
    print(x, end = ' ')
```

Ans.

```
a b c
```

270. *What is the output of the following ?*

```
d = {0: 'a', 1: 'b', 2: 'c'}
for x in d.values():
    print(d[x])
```

Ans. The above code will produce Error as dictionary's values cannot be used as keys *i.e.,* inside square brackets.

271. *What is the output of the following ?*

```
d = {0, 1, 2}
for x in d.values():
    print(x)
```

Ans. The above code will produce **error** as d stated above is not correct form of dictionary, which has keys and values.