Very Short Answer Questions

121. *What are tokens in Python ? How many types of tokens are allowed in Python ? Exemplify your answer.* [Textbook Q. 1, Chapter 1 (Type A)]

Ans. The smallest individual unit in a program is known as a Token or a lexical unit.

Python has following types of tokens :

(*i*) Keywords (*ii*) Identifiers (Names) (*iii*) Literals (*iv*) Operators (*v*) Punctuators

122. *What is the difference between a keyword and an identifier ?*

Ans. **Keyword** is a special word that has a special meaning and purpose. Keywords are reserved and are a few. *For example,* if, elif, else etc., are keywords.

Identifier is the user-defined name given to a part of a program *viz.* variable, object, function etc. Identifiers are not reserved. These are defined by the user but they can have letters, digits and a symbol underscore. They must begin with either a letter or underscore. *For instance, _chk, chess, trial* etc., are identifiers in Python.

123. *What are literals in Python ? How many types of literals are allowed in Python ?*

Ans. Literals mean constants *i.e.,* the data items that never change their value during a program run.

Python allows *five* types of literals :

(*i*) String literals (*ii*) Numeric literals

(*iii*) Boolean literals (*iv*) Special Literal *None*

(*v*) Literal Collections like tuples, lists etc.

124. *How many types of strings are supported in Python ?*

Ans. Python allows two string types :

(*i*) *Single line Strings* Strings that are terminated in single line

(*ii*) *Multiline Strings* Strings storing multiple lines of text.

125. *Can non-graphic characters be used and processed in Python ? How ? Give examples to support your answer.* [Textbook Q. 4, Chapter 1 (Type A)]

Ans. In Python strings, we can include non-graphic characters through escape sequences.

Escape sequences are predefined codes that begin with a backslash(\) following by a special character or a group of characters. Each escape sequence is treated as a single character.

Some commonly used escape sequences are :

\n newline character \t tab character \' single quote

\" double quotes \\ slash character

126. *Is the following statement valid ? Why ?*

```
>>> a = "a"
```

Ans. Yes, the above statement if fully valid in Python where a variable namely ***a*** is storing/referring to a string "**a**".

127. *Is the following statement valid ? Why ?*

```
>>> "a" == a
```

Ans. The above statement may or may not give error.

If variable ***a*** has been defined earlier, the above statement will give no error and produce a result.

But if variable ***a*** has not been defined earlier, the above statement will give NameError.

128. *Out of the following, find those identifiers, which cannot be used for naming Variables or Functions in a Python program :* [Textbook Q. 5, Chapter 1 (Type A)]

```
Price*Qty, class, For, do,
4thCol, totally, Row31, _Amount
```

Ans. Following identifiers cannot be used for naming variables /functions in a Python program :

```
Price*Qty, class, 4thCol,
```

129. *How are floating constants represented in Python ? Give examples to support your answer.* [Textbook Q. 6, Chapter 1 (Type A)]

Ans. Floating point literals or real literals or floats represent real numbers and are written with a decimal point dividing the integer and fractional parts to represent numbers having fractional parts.

These can be written either in :

Fractional form *e.g.,* – 13.0, .75, 7. etc.

or in

Exponent form *e.g.,* 0.17E5, 3.E2, .6E4 etc.

130. *How are string-literals represented and implemented in Python ?*

[Textbook Q. 7, Chapter 1 (Type A)]

Ans. A string literal is a sequence of characters surrounded by quotes (single or double or triple quotes).

String literals can either be **single line strings** or **multi-line strings**.

131. *What is None literal in Python ?*

Ans. Python has one special literal called **None**.

The **None** literal is used to indicate something that has not yet been created. It is a legal empty value in Python. None literal can be applied to all types.

132. *Consider the following code. What output will it produce and why ?*

```
>>> rec = { }
>>> rec == None
```

Ans. The above code will produce output as **False**. The reason is that **rec** is an existing empty dictionary and **None** is something which has not been created as yet. Hence the result.

133. *What is an expression and a statement ?* [Textbook Q. 9, Chapter 1 (Type A)]

Ans. **Expressions** are any legal combination of symbols that represents a value.

Statements are programming instructions.

134. *What all components can a Python program contain ?* [Textbook Q. 10, Chapter 1 (Type A)]

Ans. A python program contains various components such as :

Expressions, Statements, Comments, Functions, Block(s) or suite(s) etc.

135. *What are variables ? How are they important for a program ?*

[Textbook Q. 11, Chapter 1 (Type A)]

Ans. Variables represent labelled storage locations, whose values can be manipulated during program run.

For various types of data manipulation requirements, variables play an important role.

136. *Which data types of Python handle Numbers ?*

Ans. Python provides following data types of handle numbers (version 3.x) :

(*i*) Plain integers (*ii*) Long integers

(*iii*) Boolean (*iv*) Floating-point numbers

(*v*) Complex numbers

137. *Why is Boolean considered a subtype of integers ?*

Ans. Boolean values *True* and *False* internally map to integers *1* and *0*. That is, internally *True* is considered equal to 1 and *False* as equal to 0 (zero). When 1 and 0 are converted to Boolean through *bool( )* function, they return *True* and *False*. That is why Booleans are treated as a subtype of integers.

138. *What is the role of comments and indentation in a program ?*

Ans. Comments provide explanatory notes to the readers of the program. Compiler or interpreter ignores the comments but they are useful for specifying additional descriptive information regarding the code and logic of the program.

Indentation makes the program more readable and presentable. Its main role is to highlight nesting of groups of control statements.

139. *What is a statement ? What is the significance of an empty statement ?*

Ans. A statement is an instruction given to the computer to perform any kind of action.

An empty statement is useful in situations where the code requires a statement but logic does not. To fill these two requirements simultaneously, empty statement is used.

Python offers **pass** statement as an empty statement.

140. *How many integer types are supported by Python ? Name them.*

[Textbook Q. 14, Chapter 1 (Type A)]

Ans. Python provides following types of integer types :

(*i*) Integers (signed),

(*ii*) Booleans.

141. *What is entry controlled loop ? Which loop is entry controlled loop in Python ?*

[Textbook Q. 18, Chapter 1 (Type A)]

Ans. The loop which tests a condition before entering into the loop, is called an entry controlled loop. In Python, **while** is an entry controlled loop.

142. *Below are some segments of code, each with a part coloured. 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*)
```
if temp < 32 :
    print("Freezing")
```

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

(*iii*)
```
M = [ ]
for i in range (3) :
    M . append(i)
    print(M)
```

(*iv*)
```
L = [ 'Hiya', 'Zoya', 'Preet' ]
n = len (L)
if 'Donald' in L[1 : n] :
    print(L)
```

(*v*)
```
if n % 2 == 0 :
    print("Freezing")
```

Ans. (*i*) Boolean (*ii*) String (*iii*) List (*iv*) Integer (*v*) Boolean

143. *Find and write the output of the following python code :* [CBSE Sample Paper 2019-20]

```
x = "abcdef"
i = "a"
while i in x:
    print(i, end = "")
```

Ans. aaaaaa---

keeps printing a's endlessly OR infinite loop

144. *What is the internal structure of Python strings ?* [Textbook Q. 1, Chapter 2 (Type A)]

Ans. Strings in Python are stored as individual characters in contiguous locations, with two-way index for each location.

145. *Discuss the utility and significance of Lists, briefly.* [Textbook Q. 3, Chapter 2 (Type A)]

Ans. A list is a standard data type of Python that can store a sequence of values belonging to any type.

Also, lists are mutable types and support item assignment and thus are useful for applications where element changes are to be stored.

146. *What do you understand by mutability ? What does "in place" task mean ?* [Textbook Q. 4, Chapter 2 (Type A)]

Ans. Mutability means that in the same memory address, new value can be stored as and when needed.

The term "in place" means the place of storage for the value remains the same but the stored value changes.

147. *What's a [1 : 1] if a is a string of at least two characters ? And what if string is shorter ?* [Textbook Q. 6, Chapter 2 (Type A)]

Ans. It will be an empty string ''.

In any case, a[1:1] will return an empty string ?

148. *What will be the output of following code if a = "abcde".*

```
>>> a[1:1] == a[1:2]
>>> type (a[1:1]) == type (a[1:2])
```

Ans. The output produced by given code will be :

```
False
True
```

149. *What are the two ways to add something to a list ? How are they different ?* [Textbook Q. 7, Chapter 2 (Type A)]

Ans. The two ways of adding elements to a list are :

(*i*) **append( )** – adds one item in the end

(*ii*) **insert( )** – adds one item at the specified position

150. *What are the two ways to remove something from a list? How are they different ?* [Textbook Q. 8, Chapter 2 (Type A)]

Ans. The two ways of deleting elements to a list are :

(*i*) **remove( )** – removes one item, its first occurrence, from the list.

(*ii*) **pop( )** – removes one item from the specified position.

151. *What is the difference between a list and a tuple ?* [Textbook Q. 9, Chapter 2 (Type A)]

Ans. Lists are mutable sequence types while tuples are immutable sequence types of Python.

152. *Discuss the utility and significance of Tuples, briefly.* [Textbook Q. 11, Chapter 2 (Type A)]

Ans. Tuples are immutable sequence types of Python and are used in situations where items in a sequence must not change.

153. *If a is (1, 2, 3)* [Textbook Q. 12, Chapter 2 (Type A)]

(*a*) what is the difference (if any) between a * 3 and (a, a, a) ?

(*b*) is a * 3 equivalent to a + a + a ?

(*c*) what is the meaning of a[1:1] ?

(*d*) what's the difference between a[1:2] and a[1:1] ?

Ans.

(*a*) Expression **a * 3** will give a tuple while **(a, a, a)** will give a nested tuple :

```
>>> a = (1, 2, 3)
>>> a * 3
(1, 2, 3, 1, 2, 3, 1, 2, 3)
>>> (a, a, a)
((1, 2, 3), (1, 2, 3), (1, 2, 3))
```

(*b*) Yes

(*c*) Empty tuple ( )

(*d*) a [1:2] will give a tuple with one item while a[1:1] will give an empty tuple.

154. *What is the difference between (30) and (30,) ?* [Textbook Q. 13, Chapter 2 (Type A)]

Ans. (30) is an integer value while (30,) is a tuple.

155. *Why is a dictionary termed as an unordered collection of objects ?* [Textbook Q. 14, Chapter 2 (Type A)]

Ans. A dictionary is not a sequence type and it accesses its elements with the help of key instead of its index/position.

Hence it is an unordered collection of objects.

156. *What type of objects can be used as keys in dictionaries ?* [Textbook Q. 15, Chapter 2 (Type A)]

Ans. Immutable types

157. *Though tuples are immutable type, yet they cannot always be used as keys in a dictionary. What is the condition to use tuples as a key in a dictionary ?* [Textbook Q. 16, Chapter 2 (Type A)]

Ans. Tuples cannot be used as keys of dictionary if these contain any mutable type element such as of list type.

Tuples can be used as keys only they contain all elements of immutable types.

158. *How is del D and del D[<key>] different from one another if D is a dictionary ?* [Textbook Q. 18, Chapter 2 (Type A)]

Ans. The statement **del D** will remove the complete dictionary object, while **del D[key]** will remove only one entry from dictionary D.

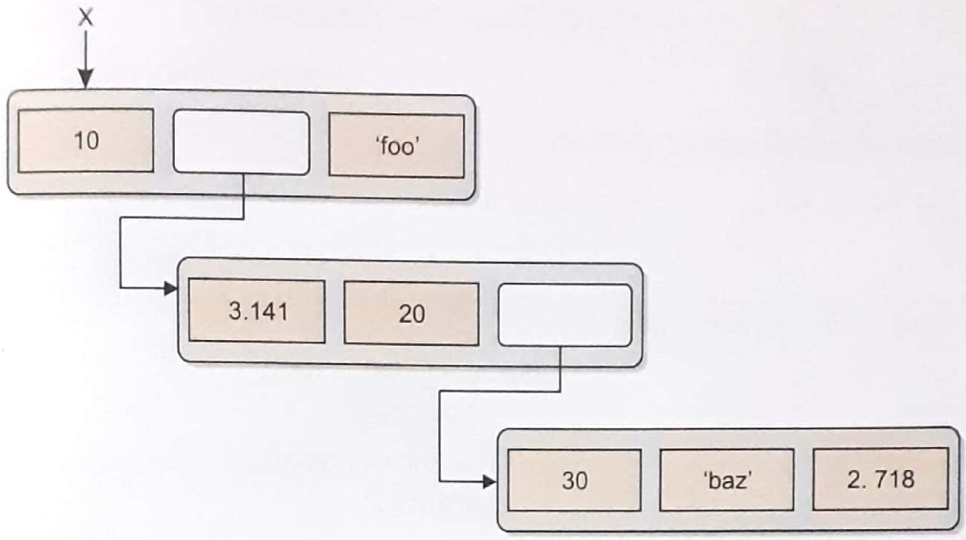
159. *What is the slice expression that gives every third character of string St, starting with the last character and proceeding backward to the first ?*

Ans. `St[::-3]`

160. *Consider the following nested list definition and then answer the questions based on this*

```
x = [10, [3.141, 20, [30, 'baz', 2.718]], 'foo']
```

A schematic for this list is shown below :



(*a*) *What is the expression that returns the 'z' in 'baz' ?*

(*b*) *What expression returns the list ['baz', 2.718] ?*

Ans. (*a*) x[1][2][1][2] (*b*) x[1][2][1:]

161. *What will be the output produced by following code ?*

```
>>> grade1 = 80
>>> grade2 = 90
>>> average1 = (grade1 + grade2)/2
>>> average2 = grade1 + grade2/2
>>> average1 , average2
```

(*a*) 85.0, 125.0 (*b*) (85.0, 125.0) (*c*) 85.0, 85.0 (*d*) (125.0, 85.0)

Ans. (*b*) (85.0, 125.0)

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

```
>>> a = 2**(3**2)
>>> b = (2**3)**2
>>> c = 2**3**2
>>> a, b, c
```

Ans. `512, 64, 512`

163. *Which of the following is valid arithmetic operator in Python ?*

(*i*) // (*ii*) ? (*iii*) < (*iv*) and [CBSE Sample Paper 2019-20]

Ans. (*i*) //

164. *Write the type of tokens from the following : (i)* ***if*** *(ii)* ***roll_no.***

[CBSE Sample Paper 2019-20]

Ans. (*i*) Keyword (*ii*) Identifier

165. *What will the following expression evaluate to ?*

```
>>> 8/4/2, 8/(4/2), (8/4)/2
```

Ans. `(1.0, 4.0, 1.0)`

166. *What is the output of the following expression ?*

```
print([4.00/(2.0+2.0),4.00/2.0+2.0 , 4.00/2.0 / 2.0])
```

Ans. `[1.0, 4.0, 1.0]`

167. *What is the output of the following expression ?*

```
print([4.00/(2.0+2.0), 4.00/2.0+2.0 ])
```

Ans. `[1.0, 4.0]`

168. *What is the output of the following codes ?*

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

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

Ans.

(*a*) `['im', 'ur']`

(*b*) `['im', 'ur']`

169. *What will the following code produce ?*

(*a*)
```
L = ['im', 'ur']
L1 = range(len(L))
for i in L1 :
    L[i] = L[i].upper()
print(L)
```

(*b*)
```
T = ('im', 'ur')
T1 = range(len(T))
for i in T1 :
    T[i] = T[i].upper( )
print (T)
```

Ans.

(*a*) `['IM', 'UR']`

(*b*) The above code will produce error as Item assignment is not possible in tuples.

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

(*a*)
```
t = (1, "a", 9.2)
t[0] = 6
```

(*b*)
```
t = [1, "a", 9.2]
t[0] = 6
```

(*c*)
```
t = [1, "a", 9.2]
t[4] = 6
```

(*d*)
```
t = 'hello'
t[0] = "H"
```

Ans.

(*a*) Item assignment not possible for tuples as tuples are immutable types.

(*b*) No error.

(*c*) Error, Item being assigned to an invalid index (index out of range).

(*d*) Item assignment not possible for strings as strings are immutable types.

171. *Name the function/method required to :* [Textbook Q. 12, Chapter 2 (Type B)]

(*i*) *check if a string contains only uppercase letters.*

(*ii*) *gives the total length of the list.*

Ans. (*a*) isupper( )

(*b*) len( )

172. *Find the errors in following function definitions :*

(*a*)
```
def main( )
    print ("hello")
```

(*b*)
```
def func2() :
    print (2 + 3)
```

(*c*)
```
def compute( ) :
    print (x * x)
```

(*d*)
```
square (a)
    return a * a
```

[Textbook Q. 4, Chapter 3 (Checkpoint 3.1)]

Ans.

(*a*) Colon(:) missing in the function header.

(*b*) If indentation is of four spaces, then No error, else indentation error.

(*c*) If indentation is of four spaces, then No error, else indentation error.

(*d*) Keyword **def** missing and no colon (:) at the end of function header.

173. *Consider a function with following header :*

```
def info(object, spacing = 10, collapse = 1):
```

Here are some function calls given below. Find out which of these are correct and which of these are incorrect stating reasons ?

For correct function call statements, specify the argument values too.

```
a. info( obj1 )
b. info(spacing = 20)
c. info( obj2, 12 )
d. info( obj11, object = obj12)
e. info( obj3, collapse = 0)
f. info( )
g. info(collapse = 0, obj3 )
h. info( spacing = 15, object = obj4 )
```

Ans.

(*a*)	Correct	***obj1*** is for positional parameter **object** ; **spacing** gets its default value of ***10*** and **collapse** gets its default value of ***1***.
(*b*)	Incorrect	Required positional argument (object) missing ; required arguments cannot be missed.
(*c*)	Correct	Required parameter **object** gets its value as ***obj2*** ; **spacing** gets value ***12*** and for skipped argument **collapse**, default value ***1*** is taken.
(*d*)	Incorrect	Same parameter object is given multiple values – one through positional argument and one through keyword(named) argument.
(*e*)	Correct	Required parameter **object** gets its value as ***obj3*** ; **collapse** gets value ***0*** and for skipped argument **spacing**, default value ***10*** is taken.
(*f*)	Incorrect	Required parameter **object's** value cannot be skipped.
(*g*)	Incorrect	Positional arguments should be before keyword arguments.
(*h*)	Correct	Required argument **object** gets its value through a keyword argument.

174. *Write the term suitable for following descriptions :* [Textbook Q. 13, Chapter 3 (Type A)]

(*a*) *A name inside the parentheses of a function header that can receive a value.*

(*b*) *An argument passed to a specific parameter using the parameter name.*

(*c*) *A value passed to a function parameter.*

(*d*) *A value assigned to a parameter name in the function header.*

(*e*) *A value assigned to a parameter name in the function call.*

(*f*) *A name defined outside all function definitions.*

(*g*) *A variable created inside a function body.*

Ans.

(*a*) Parameter (*b*) Named argument (*c*) Argument

(*d*) Default value (*e*) Named/keyword arguments

(*f*) Global Variable (*g*) Local Variable

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

```
a = 10
def call():
    global a
    a = 15
    b = 20
    print(a)
call()
```

Ans. 15

176. *What do you understand by flow of execution ?* [Textbook Q. 3, Chapter 3 (Type A)]

Ans. The *Flow of Execution* refers to the order in which statements are executed during a program run. It includes all the statements that have been executed, from the __main__ section to statements inside the functions called.

177. *What will the following function return ?* [Textbook Q. 3, Chapter 3 (Type B)]

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

Ans. **None** object will be returned. (no return statement)

178. *What will the following function print when called ?* [Textbook Q. 4, Chapter 3 (Type B)]

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

Ans. Nothing will be printed (print( ) statement is unreachable)

179. *What is the use of file __init__.py in a package even when it is empty ?* [Textbook Q. 5, Chapter 4 (Type A)]

Ans. In order for a folder containing different modules *i.e.*, .py files, to be recognized as a package, a special file namely __init__.py must also be stored in the folder, even if it is empty file.

The presence of the file **__init__.py** in a folder, makes it importable package (*i.e.*, can be imported using **import** command).

180. *How are following import statements different ?* [Textbook Q. 7, Chapter 4 (Type A)]

(*a*) import X (*b*) from X import * (*c*) from X import a, b, c

Ans.

(*a*) It imports all the defined functions and variables of module **X** and makes available to current program with **namespace X**.

(*ii*) It imports all the defined functions and variables of module **X** and makes available to current program within the namespace of current program.

(*iii*) It imports only functions/definitions of *a*, *b* and *c* of module **X** and makes available to current program within the namespace of current program.

181. *Name the Python Library modules which need to be imported to invoke the following functions :*

(*i*) *log*( ) (*ii*) *pow*( ) [Textbook Q. 11, Chapter 4 (Type A)]

Ans. (*i*) math (*ii*) math