

Q3 (i) Features of Python:-

- ① Easy to learn, write and more readable.
- ② It is interpreted not compiled.
- ③ It is open-source language under GNU public License.
- ④ Python program \rightarrow Bytecodes \rightarrow Python virtual machine \rightarrow machine language.
python is platform independent.

Scanned with CamScanner

papergrid

Date: / /

Byte codes - some low level language to be understood by some virtual machine

- ⑤ Free
- ⑥ Variable declaration do not require type.

Scanned with CamScanner

(ii)

(a)

$$5 \% 10 \text{ and } -25 > 1 * 8 // 5$$

Ans. False

(b)

$$(7 - 4 * 2) * 10 / 5 * * 2 + 15$$

Ans. 14.6

(c)

$$7 * * 2 // 9 \% 3$$

Ans

2



Scanned with CamScanner

Q1. Python Strings :

String is a collection of characters, in python strings are immutable, i.e. we cannot change a string.

Defination / declaration :- we can

use 'abc', "abc", """ abc """ to represent string object.

Gg :-
>>> str = 'abc def'
>>> str
'abc def'

String Indexing :- In python we can use positive indexes as well as negative indexes to access individual character of string.

0	1	2	3	4	5	6
a	b	c	d	e	f	
-7	-6	-5	-4	-3	-2	-1

Example :

```
>>> str[0]
```

a

```
>>> str[-1]
```

```
>>> len(str) # to display length  
of string
```

7

String Slicing :- To display only part of string, we can use slicing.

Syntax :-

```
>>> str[<starting Index> : <one more  
than the ending Index>]
```

Ex:- For printing def we can use the below slicing.

```
>>> str[4:7]
```

def

```
>>> str[:7]
```

abc def

```
>>> str[2:]
```

c def

Using + operator for string concatenation

```
>>> "abc" + "def"
```

o/p abcdef

Using * operator to repeat a string

```
>>> '*' * 5
```

Q2. Lists:- A list is a heterogeneous collection of python objects, it can also be indexed, sliced like string, but lists are mutable (i.e. we can change elements of list). It is represented in square brackets.

```
>>> list1 = [1, 2, 3, 4, 4]
```

```
>>> list2 = ['a', 'b', 1, 2, 4.5, True]
```

```
>>> list1[0] = 10
```

```
>>> list1  
[10, 2, 3, 4, 4]
```

Positive Index

0	1	2	3	4
1	2	3	4	4

Negative Index

-5 -4 -3 -2 -1

```
>>> list1[1:]  
[2, 3, 4, 4]
```

```
>>> len(list1)  
5
```


concatenate 2 list . . .

```
>>> L1 = [1, 2, 3]
```

```
>>> L2 = [3, 2, 1]
```

```
>>> L1 + L2
```

```
[1, 2, 3, 3, 2, 1]
```

Removing Elements from list

```
>>> L1 = [1, 2, 3, 4]
```

```
>>> L1.pop() # pops the last  
4 element
```

```
>>> L1
```

```
[1, 2, 3]
```

```
>>> L1.pop(2) # pop element  
at index 1.
```

```
2
```

```
>>> L1
```

```
[1, 3]
```

Remove function :- Removes the first instance of the element from list.

```
>>> L2 = [1, 2, 1, 4, 5, 1]
```

```
>>> L2.remove(1)
```

```
>>> L2  
[2, 1, 4, 5, 1]
```

• count all instances of element

```
>>> L2.count(1) # count instances  
2 of 1 in list
```

• delete a list

```
>>> del L2
```

Add elements in list (add at end)

```
>>> L3 = [1, 2, 3]
```

```
>>> L3.append(4)
```

```
>>> L3
```

```
[1, 2, 3, 4]
```

To copy a list

```
>>> L4 = L3.copy()
```


List can contain other lists

```
>>> L1 = [1, 2, 3]
```

```
>>> L2 = [4, 5, 6]
```

```
>>> L3 = [L1, L2]
```

```
>>> L3
```

```
[ [1, 2, 3], [4, 5, 6] ]
```

```
>>> len(L3)
```

```
2
```

```
>>> L3[0]
```

```
[1, 2, 3]
```

```
>>> L3[1]
```

```
[4, 5, 6]
```

```
>>> L3[0][1]
```

```
2
```

```
>>> L3.append("abc")
```

```
>>>
```

```
[ [1, 2, 3], [4, 5, 6], 'abc' ]
```

```
>>> L3[2][1]
```

```
'b'
```