iNeuron

Python

Interview Questions

1. Which of the statements about dictionary values if false?
a) More than one key can have the same value
b) The values of the dictionary can be accessed as dict[key]
c) Values of a dictionary must be unique
d) Values of a dictionary can be a mixture of letters and numbers
Answer: c
Explanation: More than one key can have the same value.

2. If a is a dictionary with some key-value pairs, what does a.pop item() do?
a) Removes an arbitrary element
b) Removes all the key-value pairs
c) Removes the key-value pair for the key given as an argument
d) Invalid method for dictionary
Answer: a
Explanation: The method pop item() removes a random key-value pair.

3. Name the important escape sequence in Python.
Ans- Some of the important escape sequences in Python are as follows:

- \\: Backlash
- \': Single quote
- \": Double quote
- \f: ASCII from feed
- \n: ASCII linefeed
- \t: ASCII tab
- \v: Vertical tab

4. What is a list?

Ans- A list is a in built Python data structure that can be changed. It is an ordered sequence of elements and every element inside the list may also be called as item. By ordered sequence, it is meant that every element of the list that can be called individually by its index number. The elements of a list are enclosed in square brackets [].

```
In [1]: #create a list
        x=['APPLE',9033,'MANGO',23444]
        print(x)

        ['APPLE', 9033, 'MANGO', 23444]
```

5. How would you access the element of the following list?

```
In [7]: #accesing items  from a list
        x=['china','pakistan','india','nepal']
        print(x[3])
        print(x[0])
        print(x[2])

        nepal
        china
        india
```

```
n [11]: z=['cricket','hockey','football','china','pakistan','india','nepal']
        print(z[:5])
        print(z[2:5])
        print(z[1:])

        ['cricket', 'hockey', 'football', 'china', 'pakistan']
        ['football', 'china', 'pakistan']
        ['hockey', 'football', 'china', 'pakistan', 'india', 'nepal']
```

6. Concatenate the two strings.

```
In [1]: #concatenate the two strings
        list1=["i love python"]
        list2=["only"]

        result=list1+list2
        print(result)

        ['i love python', 'only']
```

7. What is the difference between append () and extend () function for lists?

Ans- The append () function allows you to add one element to a list whereas extend () allows you to add more than one element to the list.

```
In [3]: #to add  an item to the list
        thislist=['apple','mango','cherry','blackberry','pineapple']
        thislist.append('orange')
        print(thislist)

['apple', 'mango', 'cherry', 'blackberry', 'pineapple', 'orange']
```

```
In [4]: x=['apple','mango','pineapple','kiwi']
        y=['banana','blueberry','cherry']
        y.extend(x)
        print(y)

['banana', 'blueberry', 'cherry', 'apple', 'mango', 'pineapple', 'kiwi']
```

8. How the format method works?

Ans- The format method works by putting a period directly after the ending string quotation, followed by the keyword “format”. Within the parenthesis after the keyword are the variables that will be injected into the string. No matter what data type it is, it will insert it into the string in the proper location, which brings up the question, how does it know where to put it? That’s where the curly brackets come in to play. The order of the curly brackets is the same order for the variables within the format parenthesis. To include multiple variables in one format string, you simply separate each by a comma. Let’s check out some examples:

```
txt = "For only {price:.2f} dollars!"
print(txt.format(price = 49))

For only 49.00 dollars!
```

9. How the strings are stored?

Ans- When a computer saves a string into memory, each character within the string is assigned what we call an “index.” An index is essentially a location in memory. Think of an index as a position in a line that you’re waiting in at the mall. If you were at the front of the line, you would be given an index number of zero. The person behind you would be given index position one. The person behind them would be given index position two and so on.
Q8. Why python is an Object-Oriented programming language?
Ans- Python is an object-oriented (OO) programming language. Unlike some other object-oriented languages, however, Python doesn’t force you to use the object-oriented paradigm exclusively: it also supports procedural programming, with modules and functions, so that you can select the best paradigm for each part of your program. The object-oriented paradigm helps you group state (data) and behaviour(code) together in handy packets of functionality. Moreover, it offers some useful specialized mechanisms covered in this chapter, like inheritance and special methods. The simpler procedural approach, based on modules and functions, may be more suitable when you don’t need the pluses1 of object-oriented programming. With Python, you can mix and match paradigms.

10. What Are Python’s Technical Strengths?
Ans- Naturally, this is a developer’s question. If you don’t already have a programming background, the language in the next few sections may be a bit baffling—don’t worry, we’ll explore all of these terms in more detail as we proceed through this book. For developers, though, here is a quick introduction to some of Python’s top technical features.

11. How Does Python Stack Up to Language X?
Ans- Finally, to place it in the context of what you may already know, people sometimes compare Python to languages such as Perl, Tcl, and Java. This section summarizes common consensus in this department.

I want to note up front that I'm not a fan of winning by disparaging the competition—it doesn't work in the long run, and that's not the goal here. Moreover, this is not a zero sum game—most programmers will use many languages over their careers. Nevertheless, programming tools present choices and tradeoffs that merit consideration. After all, if Python didn't offer something over its alternatives, it would never have been used in the first place.

12. How python is a Rapid Prototyping?
Ans- To Python programs, components written in Python and C look the same. Because of this, it's possible to prototype systems in Python initially, and then move selected components to a compiled language such as C or C++ for delivery. Unlike some prototyping tools, Python doesn't require a complete rewrite once the prototype has solidified. Parts of the system that don't require the efficiency of a language such as C++ can remain coded in Python for ease of maintenance and use.

13. What is the function of interactive shell?
Ans- The interactive shell stands between the commands give by the user and the execution done by the operating system. It allows users to use easy shell commands and the user need not be bothered about the complicated basic functions of the Operating System. This also protects the operating system from incorrect usage of system functions.

14. What does the pop() function do?

Ans- The pop function can be used to remove an element from a particular index and if no index value is provided, it will remove the last element. The function returns the value of the element removed.

```
#the pop method removes the specified index
z=['apple','guava','cherry','sugarcane']
print('before changing')
print(z)
z.pop(3)
print('after changing')
print(z)
```

```
before changing
['apple', 'guava', 'cherry', 'sugarcane']
after changing
['apple', 'guava', 'cherry']
```

```
#if you dont specify the index then this method removes the last item
x=['apple','mango','blueberry','cherry','banana','kiwi']
x.pop()
print(x)
```

```
['apple', 'mango', 'blueberry', 'cherry', 'banana']
```

15. Is there any method to extend a list?

```
x=[23,344,566,78,6,788,99,890]
y=[234,67,89,66,333,223]
y.extend(x)
print(y)
```

```
[234, 67, 89, 66, 333, 223, 23, 344, 566, 78, 6, 788, 99, 890]
```

```
x=[223,556,778,788]
z=[234,4]
z.extend(x)
print(z)
```

```
[234, 4, 223, 556, 778, 788]
```

16. Is there any method to clear the contents of a list?
Ans- Contents of a list can be cleared using clear () function.

```
#clear(removes all the elements from a list)
z=['apple','mango','cherry']
z.clear()
print(z)
```

```
[]
```

17. How to insert the item in a list.
Ans-

```
#inserting item in a list
x=['apple','mango','cherry','banana','guava']
x.insert(2,'blueberry')
print(x)
```

```
['apple', 'mango', 'blueberry', 'cherry', 'banana', 'guava']
```

18. How to copy the elements of a list?

```
#copy('returns a copy of a list')
x=['apple','mango','cherry','banana','blueberry']
x1 = x.copy()
print(x1)
```

```
['apple', 'mango', 'cherry', 'banana', 'blueberry']
```

-

19. When would you prefer to use a tuple or list?
Ans-Tuples and lists can be used for similar situations but tuples are generally preferred for collection of heterogenous datatypes whereas list are considered for homogeneous data types. Iterating through a tuple is faster than iterating through list. Tuples are idle for storing values that you don't want to change. Since Tuples are immutable, the values within are write-protected.

20. How can you create a tuple?

```
In [1]: thistuple = ("apple", "banana", "cherry")
        print(thistuple)

        ('apple', 'banana', 'cherry')
```