**Questions Asked in Interview to Senior Python Developers**

**Introduction**:

Python developers normally come from various specializations like QA/Test Automation, Web/RESTful Application Development, Bot creation, Game Development, Statistical and Scientific application development, etc. It would be quite difficult for an interviewer to understand the point of view of the developer from all of these fields. However, in order to understand the strength of a developer in python, we may look by asking some questions that are basic and each of these developer needs to know in order to use Python successfully and in an optimized way. In order to know how much the developer knows about her/his own field of interest, one may also ask some questions that are specific to a certain field and yet almost every interviewer can easily understand in a given short span of time.

**Query the Basics**:

It would be nice to start the interview with some basic questions on python. That would put both interviewee and interviewer in the groove. Plus, it would also reflect the interviewee's understanding of the language (Python) and its usage. It should be kept in mind that the interviewer's job is to select candidates, not eliminate them, so starting with some questions like the following would be nice. I have been led this way quite a few times.

1. What data structures are available in python natively? (expected answer: strings, numeric values, lists, tuples, dictionaries, objects, etc).

Basically, one may rephrase the above questions as “What data types are available in python?”.

2. Which of these data structures are mutable and which ones are immutable?

3. What does the pass statement do in Python?

4. How do you find the type of a variable in Python?

Since Python is an object oriented language, the interviewer should also start asking basic questions about the object oriented behaviour in python. For example, the interviewer could continue with the following questions:

5. What are “new style” classes in python – and since which version (approx, since this is a detail that even a good developer may not know as it doesn't pertain to her/his work in python) was it implemented in python?

6. How is multiple inheritance handled in Python? (One of the most common questions I have come across in interviews).

At this point the interviewer could start asking some questions that are not quite basics. By now, the interviewee should also be in the groove. So one could continue with the following questions (taken from my experience at various Python interviews).

7. Can we change a key in a dictionary “in position”? If not, why? How is data stored in a dictionary in Python (Pretty common question).

8. Can we put a list as an element in tuple? If so, how would a tuple's immutability be handled (since lists are mutable, quite a lot of people get confused with this question. You can change the list in the tuple but that doesn't mean that the tuple gets changed since the tuple continues to see the list object at the same address it was before, and it has no issues if the values inside the list get changed as long as the list remains in the same place).

**Get the Candidate a Bit More Involved**:

It would be good to confuse the candidate from another angle now and see how she/he performs under a little pressure. A good senior developer will come out of the situation easily, while the less experienced ones are going to have a tough time. If the interviewee can figure these answers even with less experience than required, then I feel they may be given a try by walking them towards more troubled waters. Let's ask them about meta classes now.

9. So, what is a metaclass in Python? How would you use one in Python?

10. What is “type”? What is type(type)?

At this point most interviewers tend to go a little more deep in metaprogramming in python, and that is fine. One has to consider that time is also a factor, so may be asking one single question like the following should put the lid on the topic.

11. Normally which methods should be overridden in a metaclass in Python?

Next, the interviewer may ask 2 more questions out of the following 4 commonly asked questions. After that there should be a code writing session, and that is a part that can take anything between 15 to 25 minutes. So time is a factor, and the interview should not go over an hour by any means. Keeping it between 40 to 50 mins is fine.

12. What are generators? How do they function and how are they used in code?

13. What are iterators? How would you create one and use it in code?

14. What are decorators? Apart from a function, can anything else be a decorator? What is the criteria for an object to be a decorator? (Expected answer: it needs to be a callable, so a class can also be a decorator, if it implements its \_\_call\_\_ function). Write a short decorator to check if an argument of the targetted function(s) is/are string(s).

15. What are descriptors in python. Write a small class of your choice and implement descriptors in it.

**Code Writing Session**:

Next, the interviewer would normally ask the interviewee to write some code. In some cases I have faced challenges like the following:

16a. Write code to program a lift in a highrise building. The code should be object oriented (of course, everyone wants OO code nowadays).

16b. Write a function to sort a list of strings using the quicksort algorithm (sometimes the algorithm changes to mergesort or heapsort, and since a lot of senior developers forget these algorithms, the interviewer should assist the interviewee in recalling it. However, I feel a good developer should remember at least 2 prominent algorithms, especially quicksort, since it is quite efficient and one may need to implement it in short notice.)

16c. Write a function to create the following pattern with the asterisk character:

\*

\* \*

\* \* \*

This should be done till level n, where n is a parameter to the function. (Very popular question with my last 2 employers in Python. In fact, I have asked this quite a few times. A senior developer should be good with logic at the ground level, and this problem extracts that quality. A lot of people fail here.)

**Endgame**:

In some cases, a Senior Developer may be expected to guide a team of interns or junior developers, so they should also be capable of shouldering some responsibility in the team. Questions asked towards that end tend to be like the following:

17. Let us suppose you are handling a team with 3 members under you and you are supposed to distribute work to them and see to it that they learn from it. However, you find that one of the members is not taking things seriously and she/he tends to come late/go home earlier than usual without completing the chunk of work that is supposed to be completed. How would you handle this situation?

18. If there is a fight and animosity between 2 members of your team, how would you handle the situation?

Questions like the above tend to display the maturity of the individual in the industry. Rather than handling above mentioned situations with strong words, a more practical approach would be to talk to the erring person(s) and have them know your expectations from them at first. If they still don't obey, give them a warning, and after that simply report it to the manager if they fail to comply even after one or more warnings.

**Conclusions**:

In the above 2 pages, I have mentioned the questions that I and some of my friends have faced in positions of Senior Developer interviews. These are just a sample of them, and there are some that widely differ from the above mentioned ones. But I feel those are more of ecceptional nature because the situations during the interviews were different. The above sample could be considered to be a normalized sample for the Senior Developer position.