Questions Asked:

1) Which Version control you used?

2) Have You worked on UI side?

3) Have you worked on Database front?

4) Given List have 5 elements. that is [a, b, c, d, e].
Then print last 3 elements. How to write it?

5) Given list have only 5 elements as [a, b, c, d, e]. if "print list[10]" then what happens?

6) difference between list and tuple?

7) How to convert string to object?
(Pickle / UnPickle or eval ??)

8) how to generate random numbers in python?
using random library we can generate.
>>>import random; random.randint(1,100)
>>>import random; random.seed(10); print random.randint(1,100)
>>>import numpy as np; print np.random.randint(1,100)
>>>import random; print random.choice(range(100))
>>>import secrets; print(secrets.randbelow(100)) # only in 3.6 or above python only.

9) list of 10 elements are exists. How to merge them using comma separated values.
```
>>> a
[1, 2, 3, 4, 'hara', 6, 7]
>>>
>>> map(str,a)
['1', '2', '3', '4', 'hara', '6', '7']
>>> ",".join( map(str,a) )
'1,2,3,4,hara,6,7'
>>>
```

10) How to share global variables across modules?

11) What is closure in python? Where it is used in python?

12) How to run code in parallel in python?

13) Does python support multi threading?

13.1) What is GIL in python?

14) How to randomize items in list?

15) How to define static method?

16) How to produce List of unique elements from list of duplicate elements?

17) Counting frequency of words in given list?

18) How to convert python string into a list?

```
>>> "hara hello".split()
['hara', 'hello']
>>>
>>> list("hara hello")
['h', 'a', 'r', 'a', ' ', 'h', 'e', 'l', 'l', 'o']
>>>
```

19) How to create a transpose of matrix in python? (Outer list have two inner lists. Ex: [[1,2] [3,4])

20) When we have to use '**yield**' instead of '**return**' in function.

21) I have a list with [1,2,3] another list [4,5,6]. Output should be [1,4] , [2,5] , [3,6]. how to do that?

22) Which data structure uses divide and concur algorithm in python?

23) What are the design patterns you worked on?

24) Why should I use factory method?

25) What is the issue if I keep construction of factory in factory client code?

26) What is the problem if i keep creation logic inside the client code? What is disadvantage? Why factory method?

27) how do we debug and profile an application?

28) Do you follow test driven development?

29) What are the development approaches you follow? TDD, BDD

30) How do you write unit test cases?

31) How do you mock and what mocking frameworks you have used?

32) Monkey patching?

http://www.geeksforgeeks.org/generate-graph-using-dictionary-python/

=======

1) Swap values of a, b?
a, b = b, a

2) Reversing string how?
str=”hello”; str[::-1] #this gives reversed string

3) xyz = 1,2,3,4. what is type of xyz?
xyz is a tuple

4) st1 = "h","e","l","l","o"; st2 = "hello"; # what st1 type and what is st2 type.?
st1 is a tuple and st2 is a string type

5) I have two lists, how to convert to dictionary?
dict(zip(List1,List2)) # this code will give dictionary output. List1 is keys, List2 is values

6) What is izip?
izip is attribute of itertools module.

7) Difference between zip and izip?
izip returns iterator, and zip returns a list.

8) Generator return values should be converted to fix list, how to do it? (Generator output to fixed list?)
List(generatorReturnVal) #will give the fixed list. And generator’s iter will reach to end.

9) I want to iterate thru a list, and want to print position and value. How to do?
iterable = anyCollectionObject
for i, element in enumerate(iterable):
print i, element

10) What is OrderedDict?
Ordered dictionaries are just like regular dictionaries but they remember the order that items were inserted. When iterating over an ordered dictionary, the items are returned in the order their keys were
first added.

11) How do I import a.py to b.py and also import b.py to a.py and avoid conflict?
a) Just use import a, instead of from a import *# and import b instead of from b import *
b) Use import a as my_a and import b as my_b, instead of from a import * or from b import *.

c) Import only with in a function whenever you need.

12) Write code for generator? What is it?
• Generator is a function which can stop whatever it is doing at an arbitrary point in its body, return a value back to the caller, and, later on, resume from the point it had `frozen' and merrily proceed as if nothing had happened.
• Generator is any function containing a yield keyword is a generator function.
def generate_ints(N):
for i in range(N):
yield i

13) Syntax of generator?
• Regular functions compute a value and return it, but generators return an iterator that returns a stream of values.
• Generators can be thought of as resumable functions.
• You could later resume the function where it left off.
• Generators are a special class of functions that simplify the task of writing iterators.
• Any function containing a yield keyword is a generator function.
• Generator returns a generator object that supports the iterator protocol.

14) Write code for iterator?
There are four ways to build an iterative function:
• create a generator (uses the yield keyword)
• use a generator expression (genexp)
• create an iterator (defines __iter__ and __next__ (or next in Python 2.x))
• create a function that Python can iterate over on its own (defines __getitem__)

15) What is decorator? How to write it? Explain?
>>> def outer(some_func):
... def inner():
... print "before some_func"
... ret = some_func() # 1
... return ret + 1
... return inner
a function without actual decorator:
>>> def foo():
... return 1
>>> decorated = outer(foo) # 2
>>> decorated()
before some_func #it is additional task done along with actual function foo...

- In more proper formal way decorator is:

```
@outer
def foo():
    return 1
```

16) What will happen for below code? work or throw error?

```
class A:
    def test(a,b,c):
        print a,b,c
x = A()
x.test(1,2)
```

17) What is self? is it a keyword or something else? What is it?
There is no less verbose way. Always use self.x to access the instance attribute x. Note that unlike this in C++, self is not a keyword, though. You could give the first parameter of your method any name
you want, but you are strongly advised to stick to the convention of calling itself.

18) What is the advantage of using iterator?
• For small datasets, iterator and list based approaches have similar performance. For larger datasets, iterators save both time and space.
• The performance difference isn't great on small things, but as soon as you start cranking them out getting larger and larger sets of information you'll notice it quite quickly.
• Also, not just having to generate and then step through, you will be consuming extra memory for your pre-built item where-as with the generator expression only 1 item at a time gets made.
• "Use list to code. If necessary, re-factor using iterators" The difference is not apparent unless you have a large dataset.

19) Static methods, instance methods and class methods difference? Explain?

sno	Static Method	Class Method	Instance Method
1	A static method does not receive an implicit first argument.	If we want to write a method interacts with classes only and not instances, then go for class methods	The most used methods in classes are instance methods
2	It’s definition is immutable via inheritance.	they can be overridden by subclasses, something that's simply not possible in Java's static methods or Python's	instance is passed as the first argument to the method

		module-level functions.	
3	you can put a function inside a class but you can't access the instance of that class (this is useful when your method does not use the instance).	It can be called either on the class (such as C.f()) or on an instance (such as C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.	The first parameter is always self. You can name it anything you want, but the meaning will always be the same, and you should use self since it's the naming convention.
4		Instead of passing the instance hiddenly as a first parameter, we're now passing the class itself as a first parameter.	

20) Binary search of a string in given huge file?
A quick solution is:

```
#!/usr/bin/env python
```

22) Have you used peps function tools? What is the benefit of it? How to use it.

23) Have you used robot framework?

24) In Python same method should become Static and instance methods? How?

25) What are the modules you used in python?

26) What libraries you have used in your project.?