Python Interview Questions

1. What is duck typing in python?

Answer: In duck typing, one is concerned with just those aspects of an object that are used, rather than with the type of the object itself.
For example, in a non-duck-typed language, one can create a function that takes an object of type Duck and calls that object's walk and quack methods. In a duck-typed language, the equivalent function would take an object of any type and call that object's walk and quack methods. If the object does not have the methods that are called then the function signals a run-time error. If the object does have the methods, then they are executed no matter the type of the object, evoking the quotation and hence the name of this form of typing.

More Details:
http://en.wikipedia.org/wiki/Duck_typing

2. Write a single line python expression to reverse a list.

Answer:
```
l = range(0, 11)
print "Original List: {}".format(l)
l = l[::-1]
print "Reverse List: {}".format(l)
```

(Single line expression is highlighted in bold letters above.)

3. What would be output of following program?

```
class Profile:
        pass

p = Profile()
print type(p)
print type(Profile)
```

Answer:
```
<type 'instance'> #Type of P is an instance
<type 'classobj'>
```

4. What would be output of following program?

```
class Person(object):
        pass

p = Person()
```

```
print type(p)
print type(Person)
```

Answer:

```
<class '__main__.Person'> # Type of P is of type Person
<type 'type'>
```

5. Write a single line python expression to convert following string to lowercase.

```
s = 'The Quick Brown FOX JUMPS OVER The Lazy DOG'
```

Answer:

```
s = 'The Quick Brown FOX JUMPS OVER The Lazy DOG'
s = " ".join([x.lower() for x in s.split()])
print s
```

(Single line expression is highlighted in bold letters above.)

OR

```
s = 'The Quick Brown FOX JUMPS OVER The Lazy DOG'
s = s.lower()
print s
```

6. Write a python code using **map function** to find square of numbers between 1 to 10 (both included).

Answer:

```
def square(x):
        return x*x

print map(square, range(1, 11))
```

OR

```
print map(lambda, x: x*x, range(1, 11))
```

7. Write a python code using **reduce function** to find maximum number from a list.

```
l = [23, 45, 67, 85, 35, 12, 56, 54, 48, 38]
```

Answer:

```
print reduce(lambda x, y: x if x > y else y, l)
```

OR

```
def maximum(x, y):
        if x > y:
        return x
```

```
        else:
        return y

print reduce(maximum, l)
```

8. Write a python code using **filter function** to print list elements with age > 30 for following list:
people = [{'age': 25}, {'age': 30}, {'age': 35}, {'age': 32}, {'age': 38}, {'age': 40}]

Answer:

```
print filter(lambda x: x['age'] > 30, people)
```

9. Write a python code using **sorted function** to print sorted list using key age for following list:
people = [{'age': 25}, {'age': 30}, {'age': 35}, {'age': 32}, {'age': 38}, {'age': 40}]

Answer:

```
print sorted(people, key=lambda x: x['age'])
```

10. Write a python code to print python version.

Answer:

```
import sys

print sys.version_info
```

11. What would be output of following program?

```
#Multiple inheritance
#Method resolution order
class B:
        def foo(self):
                print "Bye!"

class B1(B):
        def foo(self):
                print "Hello!"

class B2(B):
        def foo(self):
                print "Hi!"

class E1(B1, B2):
        pass

class E2(B2, B1):
```

```
        pass

e1 = E1()
e1.foo()

e2 = E2()
e2.foo()
```

Answer:
Hello!
Hi!

12. What is Global Interpreter Lock in Python(Multithreading)?

Answer:
In Python, the **global interpreter lock**, or **GIL**, is a mutex that prevents multiple native threads from executing Python bytecodes at once. This lock is necessary mainly because Python's memory management is not thread-safe.
More Details: https://wiki.python.org/moin/GlobalInterpreterLock

13. How to set up a very basic web server serving files relative to the current directory in Python?

Answer:
Run following command in current directory:
python -m SimpleHTTPServer

14. Write a single line python code to swap values of following two variables:
a = 20
b = 10

Answer:
a, b = b, a

15. Explain memory management in Python.

Answer:

- Memory management in Python involves a private heap containing all Python objects and data structures. Interpreter takes care of Python heap and that the programmer has no access to it. The allocation of heap space for Python objects is done by Python memory manager. The core API of Python provides some tools for the programmer to code reliable and more robust program.
- Python also has a build-in garbage collector which recycles all the unused memory. When an object is no longer referenced by the program, the heap space it occupies can be freed. The garbage collector determines objects which are no longer

referenced by the program frees the occupied memory and make it available to the heap space.

- The gc module defines functions to enable /disable garbage collector:
  gc.enable() -Enables automatic garbage collection.
  gc.disable() - Disables automatic garbage collection.

16. Explain negative indexing in Python.

Answer:
Array elements in Python can be accessed using negative index. -1 will refer to last element in the array, -2 will refer to second last element in array and so on.
e. g.

```
a = [1, 2, 3, 4, 5]
print a[-1]
print a[-2]
```

Output:

```
5
4
```

17. Write a code to explain exception handling in Python.

Answer:

```
while True:
        try:
                x = int(raw_input("Enter no.          of your choice: "))
                break
        except ValueError:
                print "Oops! Not a valid number. Attempt again"
```

18. What is the difference between a tuple and a list?

Answer:
A tuple is immutable i.e. can not be changed. It can be operated on only. But a list is mutable.

tuple initialization: t = (1,2,3)
list initialization: l = [4,5,6]

19. What is use of __init__.py?

Answer:
It declares that the given directory is a package.

20. When is to use of **pass** in Python?
Answer:
pass does nothing. It is used for completing the code where we need something.

e.g.

```
def foo()
        pass

class MyClass:
        pass
```

21. Is Python Aspect Based Programming language?

Answer: Yes, there are a number of Python features which support functional programming and aspect-oriented programming (including by metaprogramming and by magic methods).
http://en.wikipedia.org/wiki/Aspect-oriented_programming
http://en.wikipedia.org/wiki/Python_%28programming_language%29
http://en.wikipedia.org/wiki/Metaprogramming
http://en.wikipedia.org/wiki/Metaobject
22. What is difference between range and xrange in Python?

Answer:
range creates a list, so if you do range(1, 10000000) it creates a list in memory with 10000000 elements.

xrange is a generator, so it evaluates lazily.

23. Write a optimized Python recursive function to print fibonacci series.

Answer:

```
cache_map = {}

def fib(n):
        if n == 0 or n == 1:
                return 1
                elif n in cache_map:
                return cache_map[n]
        cache_map[n] = fib(n-1)+fib(n-2)
        return cache_map[n]
```

24. Write factorial code for number 'n' using recursion/recursive method

```
def factorial(n):
        if n < 1:
           return 1
        else:
```

```
    return n * factorial(n-1)
```

25. Why should we not use '+' for string concatenation

Answer:
In Python the string object is immutable - each time a string is assigned to a variable a new object is created in memory to represent the new value. This contrasts with languages like perl and basic, where a string variable can be modified in place. The common operation of constructing a long string out of several short segments is not very efficient in Python if you use the obvious approach of appending new segments to the end of the existing string. Each time you append to the end of a string, the Python interpreter must create a new string object and copy the contents of both the existing string and the appended string into it. As the strings you are manipulating become large this process becomes increasingly slow.

source: http://www.skymind.com/~ocrow/python_string/

26. (Generic coding standard question) Why should we not use tab and use spaces instead for indentation of code

Answer: It becomes a nightmare when people merge code conflicts. They will spent more time in resolving code indentation than merging code.

Python Debugging Questions

1. When you import a module, how to make sure correct module is loaded when there are modules with same name in a common libraries?

Answer:
>>> import string
>>> dir(string)
['Formatter', 'Template', '_TemplateMetaclass', '__builtins__', '__doc__', '**__file__**', '__name__', '__package__', '_float', '_idmap', '_idmapL', '_int', '_long', '_multimap', '_re', 'ascii_letters', 'ascii_lowercase', 'ascii_uppercase', 'atof', 'atof_error', 'atoi', 'atoi_error', 'atol', 'atol_error', 'capitalize', 'capwords', 'center', 'count', 'digits', 'expandtabs', 'find', 'hexdigits', 'index', 'index_error', 'join', 'joinfields', 'letters', 'ljust', 'lower', 'lowercase', 'lstrip', 'maketrans', 'octdigits', 'printable', 'punctuation', 'replace', 'rfind', 'rindex', 'rjust', 'rsplit', 'rstrip', 'split', 'splitfields', 'strip', 'swapcase', 'translate', 'upper', 'uppercase', 'whitespace', 'zfill']
>>> **print string.__file__**
/usr/lib/python2.7/string.pyc

2. When to use inspect.getmembers() vs __dict__.items() vs dir()

Answer:

Source: http://stackoverflow.com/questions/6761106/inspect-getmembers-vs-dict-items-vs-dir

`dir()` allows you to customize what attributes your object reports, by defining `__dir__()`.

From the manual, if `__dir__()` is not defined:

If the object is a module object, the list contains the names of the module's attributes.

If the object is a type or class object, the list contains the names of its attributes, and recursively of the attributes of its bases.

Otherwise, the list contains the object's attributes' names, the names of its class's attributes, and recursively of the attributes of its class's base classes.

This is also what `inspect.getmembers()` returns, except it returns tuples of `(name, attribute)` instead of just the names.

`object.__dict__` is a dictionary of the form `{key: attribute, key2: atrribute2}` etc.

`object.__dict__.keys()` has what the other two are lacking.

From the docs on `inspect.getmembers()`:

getmembers() does not return metaclass attributes when the argument is a class (this behavior is inherited from the dir() function).

For `int.__dict__.keys()`, this is

```
['__setattr__', '__reduce_ex__', '__reduce__', '__class__',
'__delattr__', '__subclasshook__', '__sizeof__', '__init__']
```

To summarize, `dir()` and `inspect.getmembers()` are basically the same, while `__dict__` is the complete namespace including metaclass attributes.

3. Write a command to debug a python script.

Answer:

python -m pdb myscript.py

4. How to print global variables in python debugger shell?

Answer:
(pdb)globals()

5. How to print local variables in python debugger shell?

Answer:
(pdb)locals()

6. In python debugger shell what is the difference between s(tep) and n(ext)?
Answer:
The difference between next and step is that step stops inside a called function, while next executes called functions at (nearly) full speed, only stopping at the next line in the current function.

7. How to print source code in python debugger shell?

Answer:

(pdb)l

Type l character and press enter in debugger shell prompt

8. How to print stack trace in python debugger shell?

Answer:
(pdb)w

Type w character and press enter in debugger shell prompt

Python Programs

1. Write a program to print tomorrow's date
2. Write a program to find out whether palindrome string can be formed using user input
3. Write a program to print count of least possible squares that can be formed from a rectangle by using whole space of that rectangle
4. Write a program to print value of a key in a recursive python-dictionary
5. Write a program to merge two dictionaries
6. Write a program to print list of all *.txt files from given directory
7. Write a program to print fibonacci series
8. Write a program to find sum of digits of an integer
9. Write a program to reverse digits of an integer
10. Write a program to print frequency of each item in the given list
11. Write a program to check whether parenthesis - [], {}, () in a given string are matching or not
    a. {()}[{}()] : True
    b. [{]}({}] : False
12. Write a program to print all possible combinations of number 1,2,3,4 without repeating any number:
    1,2,3,4
    1,3,4,2
    1,4,3,2
    1,4,3,2 etc.
13. Write a Python code using list comprehension to capitalize each word in a given string
    Input: A quick brown fox jumps over the lazy dog.
    Output: A Quick Brown Fox Jumps Over The Lazy Dog.
14. Write a Python code using list comprehension to remove 'the' word from a given string
    Input: The quick brown fox jumps over the lazy dog.
    Output: quick brown fox jumps over lazy dog.
15. Write a Python code using list comprehension to return last five elements in given list in reverse order
    Input: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
    Output: [100, 90, 80, 70, 60]
16. Write a Python method which will take input as string and return True if given string has at least one capital letter, at least one small letter and one digit in it